91602
new 82-10

COMPLIMENTARY COPY
Please Credit Ontario Hydro Research if
Reproduced in Whole or in Part

ontario hydro
research division

To: Mr. J.E. Mitchell
Manager
Technical Services & Development
Program Support & Services Division

Advanced PC-Based Program for Power Quality Analysis of Variable Speed Drives

G.K. Hatanaka

An advanced PC-based program and its novel computational method for the Power Quality analysis of variable speed drives is described. The program can simulate accurately the voltage and current harmonics, both characteristic and non-characteristic, injected by variable speed drives into the supply system. Numerical validation of the program is given and results of field tests on variable speed drives are presented to demonstrate the capabilities of the program. The program is intended for end-use applications and is structured for use by non-experts.

1. Introduction

The advantages of variable speed drives (VSDs) are well-known and understood. Not so well understood are the power quality aspects of the drives which must be addressed in the overall assessment of the drive performance and cost benefits. The drives can generate considerable harmonic disturbances which can create problems for customers operating sensitive electronic equipment. On the other hand, a variable speed drive itself may be vulnerable to specific disturbances in the power system. It is, therefore, essential to have an accurate tool to properly evaluate and control these Power Quality aspects of drive performance.

This report describes the development of an advanced PC-based program and its novel computational method for the power quality analysis of VSDs. This project was performed by the Electrical Research Department for the Technical Services & Development Department of Program Support & Services Division, Energy Management Branch. Ryerson Polytechnical Institute was retained as a subcontractor to develop the program under contract to the Electrical Research Department.

job	file	date	report no.
18-140-112	816.22	December 31, 1991	91-294-K

LEGAL NOTICE

This report was prepared by Ontario Hydro and, while effort has been made to assure the accuracy and applicability of the material contained herein, neither Ontario Hydro nor any person acting on its behalf:

(a) makes any warranty, express or implied with respect to the use of any information, apparatus, method or process disclosed in this report or that such use may not infringe privately owned rights; or

(b) assumes any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, method or process disclosed in this report.

92-04130/

2. *Summary*

Computational methods have traditionally been in the frequency domain. However, these methods assume a very simple model for all drive harmonics and cannot predict the effects of non-characteristic harmonics. Assessment of VSD susceptibility to power system disturbances is also precluded in the frequency domain. On the other hand, time domain simulation techniques offer the means to accurately assess both aspects of VSD Power Quality performance. Time domain simulation packages such as the EMTP (ElectroMagnetic Transients Program) are available for computing the waveforms of a switching circuit but are not easily implemented for VSDs and are intended for experts.

The advanced program uses established time domain simulation techniques but uses a novel nodal approach which allows all switchings in the drive converter to be easily monitored and the circuit equations to be efficiently computed. This approach results in a fast solution compared to the standard state space formulation for the time-domain simulation.

The special algorithm for the program allows the dynamic simulation of an arbitrarily configured drive circuit which can incorporate feedback controls. There are two key advantages of this approach. Firstly, it can simulate non-characteristic harmonics due to non-ideal conditions and secondly, it allows the computation of the impact of power system disturbances on the drive performance.

The current version of the program only allows the computation of drive harmonics. Further work is required to expand the program's capability to evaluate drive susceptibility. The program is intended for end-use applications by customers, utilities and consultants and is structured for use by non-experts. A graphical user-interface for data input and output was developed as part of the project but this aspect of the program will be reported separately.

Details of the program structure and computational methods and formulations are covered in the Report prepared by Ryerson Polytechnical Institute, which is reproduced in the Appendix.

3. *Acknowledgements*

I gratefully acknowledge the fine work and dedication of Richard Cheung and Louis Cheng of Ryerson Polytechnical Institute throughout the course of this project. I also wish to thank Scot Hutchinson and his staff for their cooperation and assistance in conducting field tests at LASCO. I am also grateful to Bruno Pietrantonio and Don Wisenden for their excellent work in performing the field tests and reducing the data.

Approved:

A.F. Baljet
Manager
Electrical Research Department

GKH:am

Submitted:

G.K. Hatanaka
Research Engineer
System Studies Section

Appendix

Report for

Ontario Hydro Research Division
800 Kipling Avenue
Toronto, Ontario M8Z 5S4

ADVANCED PC-BASED PROGRAM FOR POWER QUALITY ANALYSIS OF VARIABLE SPEED DRIVES

Prepared by

Richard Cheung, Ph.D., P.Eng.
Principal Investigator
Electrical Engineering Department
Ryerson Polytechnical Institute

November 1991

Introduction

This report presents an advanced PC-based program for the power quality analysis of variable speed drives. The computer program can simulate accurately the voltage and current harmonics injected by variable speed drives into the supply system, particularly the non-characteristic harmonics due to the non-ideal operating conditions.

This report has been prepared by Dr. Richard Cheung of Ryerson Polytechnical Institute, for the Ontario Hydro Research Division under a contract to develop a software package for the power quality analysis of variable speed drives.

The program uses established time domain simulation techniques but employs a novel nodal approach for circuit formulation. This formulation allows all switchings in the drive converter to be easily monitored and the circuit equations to be efficiently computed. This results in a fast solution compared to the standard state space formulation for the time domain simulation. The program overcomes difficulties of existing softwares using the nodal formulation, such as the computation of several switches connected at one circuit node.

The program implements an automatic formulation algorithm. It uses straightforward data such as the nodes and values of components to formulate a network equation for an arbitrarily configured drive circuit. The algorithm eliminates the requirement of some time-domain programs to model a drive circuit using additional artificial resistors. This eliminates the numerical stability problems introduced by these resistors. A graphic user-interface has been developed for the program to enhance its usability. This is the subject of a separate report.

This report covers:

1. Selection of computational methods for the program.

2. Formulations for simulations of variable speed drives.

3. Implementation of nodal formulations in the program.

4. Numerical and field-test validations of the program.

1. Methods of Computation

First, this chapter provides a brief assessment of computational methods often used for power quality analysis of variable speed drives. Second, this chapter describes the time domain method using a novel nodal network formulation, which has been implemented in the advanced PC-based program to compute the drive harmonics.

1.1 Frequency Domain Methods

To date, it has been standard to use a frequency domain method to compute harmonics in a power system. This method has two basic formulations: linear harmonic powerflow, and iterative powerflow computation. The first is a simple linear network formulation. This can be used to effectively compute harmonics in a linear electrical network, consisting of linear impedance loads and known harmonic sources [1-2]. The second formulation requires numerical iterations and can handle the conventional model of constant power loads [3-5].

However, the frequency domain method, when employed for the drive analysis, must assume a fixed amount of characteristic harmonics injected from the drive regardless of its operating conditions. The frequency domain method, therefore, cannot accurately predict harmonics generated by a practical drive system, particularly when it operates under a non-ideal circuit condition [6].

1.2 Time Domain Methods

The harmonics injected from a variable speed drive into a power distribution system will depend on the conditions of the supply system and the drive control. The drive harmonics, both the characteristic ones of ideal operations and non-characteristic ones from unbalanced operations, can be simulated correctly using the time domain method [7].

To date, the state space technique has been the standard practice for the formulation of the time domain analysis of variable speed drives. In this formulation, each state of a drive operation representing a combination of conducting switches is modeled with one state equation, which consists of a set of first order differential equations. The state equation can be obtained by the manual manipulation of loop and node equations according to the equivalent circuit of the state being modeled. This manipulation becomes difficult in the formulation of a general drive analysis, when many states are usually encountered. A circuit-topological algorithm capable of substantially reducing this difficulty has been reported [8]. Even with this algorithm, the computational burden using the state space approach is still considerable, due to the involved formulation and solution process [9].

Alternatively, the nodal formulation technique, which was originally designed for power system transient analysis [10], can be used to analyze variable speed drives. In this technique, the components of a drive circuit are first modeled individually using the implicit trapezoidal integration method [10], and then they are assembled into one nodal equation at each node of the drive circuit. As the state of the drive operation changes, some of the nodal equations must be re-assembled.

1.3 Existing Software Implementations of Time Domain Methods.

Several time domain simulation packages are available, such as EMTP (ElectroMagnetic Transients Program) and SPICE (Simulation Package for Integrated Circuits). Most of the packages require the circuit data of drive converters to be assembled in a rigid format. This makes the analysis of a general drive very difficult. Many packages cannot compute directly, with no circuit modification, the drive configuration where several switches are connected together. If this drive is simulated, some packages require the addition of artificial resistors to separate the interconnecting switches in the converters [11]. Other methods suggest to model a switch using a resistor having two extreme values for the on- and off-state [12]. This can result in an ill-conditioned network matrix and can cause computation difficulties for both the nodal or state space formulation.

None of the existing packages provide a general accurate dynamic model for induction machines [13]. Some use the basic machine equation being a function of the rotor angle [14]. This equation is computational inefficient. Others use the classical DQ (direct-quadric) machine equation to improve the computation. The solution of this equation requires the drive terminal voltages and currents to be transformed into the same DQ reference. This transformation is non-trivial and cannot be generalized for all drive conditions. Therefore, the simulation of ac drives using these packages is complicated [13].

1.4 Improved Computational Method selected for the Program

The harmonics from variable speed drives under various operating conditions can only be calculated accurately using the time domain simulation together with post-processing using Fourier analysis. An improved nodal network formulation for time domain simulation has been developed and implemented in the advanced computer program. The formulation has overcome the major difficulties in the existing time domain packages, such as the modelling and monitoring of changes of switching states, and the modelling of ac drives. The formulation will be detailed in Chapter 2, and the software implementation will be discussed in Chapter 3.

2. Nodal Formulation for Time Domain Simulation

This chapter presents a novel nodal network formulation for power-electronic circuit analysis. The formulation has been implemented in the advanced PC-based program for the power quality analysis of variable speed drives. The following steps are employed in the formulation.

1. Model all circuit components using the implicit trapezoidal integration method.
2. Assemble all nodal network equations in a matrix form, assuming that all switches are open.
3. Determine the actual open / close position of each switch by monitoring its voltage and current.
4. Manipulate the nodal network matrix according to the actual status of all switches.
5. Formulate the LU (lower-and-upper) matrix decomposition which is enhanced with a network sparsity arrangement.

Features of the program are described below:

2.1 Nodal Formulation of Circuit Components

The circuit components are modeled individually using the implicit trapezoidal integration method [10]. The general formulation is illustrated below.

The modelling of an inductor starts from its basic equation:

$$v_L = L \frac{di}{dt} \tag{1}$$

This equation can be approximated using the implicit trapezoidal rule to give:

$$\frac{1}{2}\left[v_L(t) + v_L(t-\Delta t)\right] = \frac{L}{\Delta t}\left[i_L(t) - i_L(t-\Delta t)\right]$$

which is simplified to:

$$i_L(t) = Y_L\, v_L(t) + I_L \tag{2}$$

where $Y_L = \frac{\Delta t}{2L}$ and $I_L = i_L(t-\Delta t) + Y_L v_L(t-\Delta t)$.

Y_L is the equivalent admittance. I_L is the equivalent current source computed from the previous time step of (t-Δt). A circuit representation of this formulation is shown in Figure 1. Similarly, the nodal formulation of a capacitor is:

$$i_c(t) = Y_c v_c(t) + I_c \tag{3}$$

where $Y_c = \frac{2C}{\Delta t}$ and $I_c = -i_c(t-\Delta t) - Y_c v_c(t-\Delta t)$.

The formulation for a resistor is trivial:

$$i_R(t) = Y_R v_R(t) \tag{4}$$

where $Y_R = 1 / R$. These equations are the three basic elements used to form a node equation.

2.2 Nodal Formulation of Circuit Modules

The above three basic nodal equations can be used to formulate a module for a combination of circuit components. For example, they can be combined to represent a series connection of resistor, inductor, and capacitor as shown in Figure 2:

$$i_{RLC}(t) = Y_{RLC}\left[v_1(t) - v_0(t)\right] + I_{RLC} \tag{5}$$

where $1/Y_{RLC} = 1/Y_R + 1/Y_L + 1/Y_C$,

and $$I_{RLC} = Y_{RLC}\left[1/Y_L - 1/Y_C - 1/Y_R\right] i_{RLC} + Y_{RLC}\left[v_1(t-\Delta t) - v_0(t-\Delta t)\right] - 2Y_{RLC} v_c(t-\Delta t)$$

This equation is very useful to formulate other circuit modules, such as a three-phase tuned power filter. Its circuit representation is shown in Figure 3, and the nodal equations are:

$$\begin{aligned} i_{1RLC}(t) &= Y_{RLC}\left[v_1(t) - v_0(t)\right] + I_{1RLC} \\ i_{2RLC}(t) &= Y_{RLC}\left[v_2(t) - v_0(t)\right] + I_{2RLC} \\ i_{3RLC}(t) &= Y_{RLC}\left[v_3(t) - v_0(t)\right] + I_{3RLC} \end{aligned} \tag{6}$$

2.3 Assembly of Nodal Network Equations

Figure 4 is used to illustrate the formulation of a node equation. At node *A* with the switch open, the total current injected from the inductor and the capacitor equals zero: $i_L + i_C = 0$. Therefore, the node *A* equation is:

$$Y_L(v_A - v_s) + I_L + Y_C(v_A - v_C) + I_C = 0 \tag{7}$$

Similarly for node *B*:

$$Y_R(v_B - v_C) = 0 \tag{8}$$

These two equations in the matrix form are:

$$\begin{bmatrix} Y_L+Y_C & 0 & -Y_C & -Y_L \\ 0 & Y_R & -Y_R & 0 \end{bmatrix} \begin{bmatrix} v_A \\ v_B \\ v_C \\ v_S \end{bmatrix} = \begin{bmatrix} -I_L-I_C \\ 0 \end{bmatrix} \tag{9}$$

The program uses this method to automatically assemble the nodal equations: [Y].[v] = [I], such that the diagonal of [Y] sums all admittances connected to its node and the off-diagonal equals the negative admittance of the inter-connecting node. The program also includes several modules of standard network components and a fast algorithm to assemble a complete drive equation from these modules. Matrix computation of the drive equation uses standard LU decomposition techniques and an efficient sparse network algorithm developed for the program [15,16].

2.4 Novel Computational Algorithm

The operation of a variable speed drive involves many switchings of various combinations. This results in two difficult tasks. First, the computer must monitor the operating state of each switch from its voltage and current values. These values cannot be obtained directly from the solution of the state equation when the state space formulation is used [17]. Second, the computer must formulate a new network equation each time the operating state changes. Some existing packages use a variable resistor to model the ever-changing switch. Others, such as the EMTP require the switches to be separated by additional resistors. Either solution may cause numerical problems.

To overcome these difficulties, a simple algorithm was developed for the advanced power quality computer program. The switch voltage and current can be obtained easily from the node voltage and from the sum of the currents injected into the switch node as part of the nodal network solution. The algorithm re-organizes the network matrix to model the change of switch states. This is illustrated using Figure 4. Equation (9) which was originally obtained for the state of the

open switch can easily be manipulated to model the state of the closed switch. This is done by adding the two rows of equation (9) together:

$$\begin{bmatrix} Y_L+Y_C & Y_R & -Y_C-Y_R & -Y_L \end{bmatrix} \begin{bmatrix} v_A \\ v_B \\ v_C \\ v_S \end{bmatrix} = \begin{bmatrix} -I_L-I_C \end{bmatrix} \qquad (10)$$

Similarly, this algorithm can be used to model several switchings which occur at the same time or at the same location.

2.5 Robust Numerical Computation

The advanced power quality program uses two very robust numerical formulations. First, this program can manipulate the network equations for the change in operating states of switches, requiring no additional artificial resistors. This eliminates the numerical stability problems introduced by these resistors.

Second, all circuit components are modeled using the implicit trapezoidal integration method. This method is numerically very stable, but it may have some numerical oscillations during the switching intervals. The program implements a special algorithm, described in the literature [18], that uses the explicit Backward Euler method [15] to eliminate the possible oscillation problem during the switching interval, and the implicit formulation otherwise. Using the Backward Euler method, equation (1) becomes:

$$i_L(t) = \frac{\Delta t}{L} v_L(t) + i_L(t-\Delta t) \qquad (11)$$

This is compared to the formulation using the implicit method (Eq. 2):

$$i_L(t) = \frac{\Delta t}{2L} v_L(t) + i_L(t-\Delta t) + \frac{\Delta t}{2L}\left[v_L(t-\Delta t)\right]$$

The sampled interval for the explicit form is twice that of the implicit form. This simplifies the program since the computation of the explicit form for one interval is equivalent to two using the implicit form. This eliminates the need of an additional network matrix for the explicit form. The implicit form is used for all calculations except during the switching. When a switching occurs, the program executes the implicit matrix twice with an appropriate i_L to obtain one sample. The numerical oscillation problem is eliminated because the calculation of $i_L(t)$ in (11) does not use the previous value, $v_L(t-\Delta t)$.

2.6 Efficient Machine Model

A novel induction motor model is implemented in this program to enhance the analysis of ac drives. This model is derived from the basic motor equation:

$$\left[V_{s,r}\right] = \left[R_{s,r}\right] \cdot \left[i_{s,r}\right] + \frac{d}{dt}\left[L(\theta)\right] \cdot \left[i_{s,r}\right] \qquad (12)$$

where $[v_{s,r}]$ = motor stator and rotor voltages, $[i_{s,r}]$ = motor currents, $[R_{s,r}]$ = motor resistances, and $[L(\theta)]$ = motor inductances. These matrices are detailed in [16]. This motor equation cannot be computed efficiently because the inductance matrix $[L(\theta)]$ is a function of the constantly changing rotor angle θ. A classical way of eliminating this angle dependence is to apply a DQ (direct and quadric) transformation to both the stator and the rotor voltages and currents of (12). This results in the well known DQ motor equation:

$$\left[V_{dq}\right] = \left[R(\omega)\right] \cdot \left[i_{dq}\right] + \left[L\right] \cdot \frac{d}{dt}\left[i_{dq}\right] \qquad (13)$$

where the matrices are detailed in [16]. In particular, [L] becomes constant and $[R(\omega)]$ is a function of the slow-changing rotor speed ω. This makes the computation of (13) much faster than that using the basic motor equation. Therefore, it is standard practice to use this classical DQ motor equation for the analysis of ac drives [19]. However, the drive terminal voltages and currents must be transformed into DQ variables which are then incorporated with (13) to formulate a drive system. The combination of the drive terminal voltages and currents, which can be formulated together with (13), will change as the drive switches operate. It has been demonstrated that this formulation is dependent on the drive terminal conditions and is not trivial, particularly for implementation in a general drive computer program [20].

The conventional DQ formulation for the drive analysis can be significantly simplified if the DQ transformation is applied to the rotor variables of (12) only. This results a new stator-reference motor equation [15]:

$$\begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \\ v_{dr} \\ v_{qr} \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 & 0 \\ 0 & 0 & R_s & 0 & 0 \\ 0 & 0 & 0 & R_r & 0 \\ 0 & 0 & 0 & 0 & R_r \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \\ i_{dr} \\ i_{qr} \end{bmatrix} \begin{bmatrix} L_{ss} & L_{sm} & L_{sm} & L_1 & 0 \\ L_{sm} & L_{ss} & L_{sm} & L_2 & L_3 \\ L_{sm} & L_{sm} & L_{ss} & L_2 & L_4 \\ L_1 & L_2 & L_2 & L_r & 0 \\ 0 & L_3 & L_4 & 0 & L_r \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \\ i_{dr} \\ I_{qr} \end{bmatrix} \qquad (14)$$

where the new parameters in terms of the standard ones are:

$$L_{ss} = L_s - L_m/3, \quad L_{sm} = -L_m/3, \quad L_1 = \sqrt{2}\, L_m/\sqrt{3},$$

$$L_2 = -L_m/\sqrt{6}, \quad L_3 = L_m/\sqrt{2}, \quad L_4 = -L_3,$$

$$v_{dr} = \omega_r\left(L_3 i_{cs} - L_3 i_{bs} - L_r i_{qr}\right) \qquad (15)$$

$$v_{qr} = \omega_r\left(L_1 i_{as} + L_2 i_{bs} + L_2 i_{cs} + L_r i_{dr}\right)$$

The standard parameters are: L_s = stator self inductance, L_m = mutual inductance, L_r = rotor inductance, ω_r = rotor speed, R_s = stator resistance, and R_r = rotor resistance. This new motor equation has several features. Most notably, the stator voltages and currents are in the actual phase quantities such as v_{as} or i_{as}. Therefore, the drive terminal quantities can be combined directly with this new equation without the need to transform them into DQ quantities for every switching. Also, in this new equation, the rotor speed is a parameter, and the coefficient matrix of the derivative vector is constant. This preserves the good computational features of the standard DQ motor equation (13).

Furthermore, by exploiting the symmetric-matrix property of this new equation, a general circuit model for the induction motor is developed, as shown in Figure 5. This model is an ordinary circuit having R-L parameters, speed-dependent voltage sources, and actual *abc*-phase stator voltages and currents. Also, this model is valid for any drive terminal conditions. Using this model, an arbitrarily configured ac drive circuit can be formulated and solved automatically by the computer. This improves the conventional machine formulation which requires considerable manual manipulation.

2.7 Harmonics Computation

The computation of harmonics from drives requires post-processing of the voltage and current data obtained from the time domain simulation. The Fourier (cosine and sine) coefficients can be computed as follows:

$$a_k = \frac{2}{N} \sum_{j=1}^{N} \left[f(j) \cos\left(kj \frac{2\pi}{N} \right) \right]$$

$$b_k = \frac{2}{N} \sum_{j=1}^{N} \left[f(j) \sin\left(kj \frac{2\pi}{N} \right) \right] \qquad (16)$$

where f(j) are the simulated voltage and current data. "N" is the number of samples, and "k" is the order of harmonics. Therefore, the magnitude and the phase of each harmonic are:

$$\text{Magnitude} = \sqrt{\frac{1}{2}\left(a_k^2 + b_k^2\right)}$$

$$\text{Phase} = \tan^{-1}\left(\frac{a_k}{b_k}\right) \qquad (17)$$

3. Software Implementation using Nodal Formulation

This chapter describes the software implementation of nodal network formulations for the power quality analysis of variable speed drives. In particular, this chapter will focus on the structure of the computational part of the program. This part is written using the "Fortran" language, and the source codes is attached. The Graphical User-Interface (GUI) is developed using the "C" language to input and output data. This will be the subject of a separate report. Further work is required for converting the Fortran codes into "C" to achieve a unique coding for the program.

The following steps are involved for computing the harmonics from the drives:

1. Identify, arrange, and formulate all circuit components.
2. Assemble, and compute the network matrix.
3. Perform the LU decomposition on the network matrix.
4. Determine the states of switching in a drive operation.
5. Modify the LU-decomposed network matrix, if a change of switching state is detected.
6. Solve the LU-decomposed network matrix.
7. Repeat from Step 4, until the period of simulation is completed.
8. Compute the harmonics for the results of simulation.

These steps are described below:

3.1 Component Input Module

First, the program executes the Component Input Module to identify and to formulate all circuit components compiled in the input file by the GUI software. The module identifies the connections of a component in a drive circuit by its terminal nodes. The module then formulates the circuit components using the nodal approach and the implicit trapezoidal integration method as discussed in Sections 2.1 and 2.2. The formulation is performed according to the type of components in an order: motors, voltage sources, switches, coupled lumped RL, discrete lumped RLC, and current sources. This ordering is imposed by the arrangement of the network matrix [Y].

The nodal voltages [v] in the network equation of [Y].[v] = [I] is expressed with respect to "ground potential" in the program. Therefore, if the program is used to simulate a "floating" drive circuit, a potential reference must be provided by the user.

3.2 Network Matrix Module

Second, the program executes the Network Matrix Module. The module compiles the network matrix [Y] according to the current law: the sum of currents injected into a circuit node equals

zero (see Section 2.3). The arrangement of nodes in the network matrix is the following order: lumped components, switches, and voltage sources. This arrangement optimizes the computation of the matrix [16]. All switches are assumed to be open when the matrix is compiled.

3.3 LU Decomposition Module

Third, the program executes the LU Decomposition Module. The module implements an efficient sparse network algorithm and uses the established method for the LU (lower-and-upper) decomposition of the network matrix [15,16].

3.4 Switching Module

Fourth, the program executes the Switching Module. The module determines the switching state by monitoring the voltage, current, and control signal of each switch. For example, the current zero-crossing of a conducting SCR will be monitored to determine the change of state when it stops conducting. Similarly, the firing signal and the terminal voltages of a non-conducting SCR will be monitored to detect when it starts to conduct.

3.5 Modification Module

Fifth, the program executes the Modification Module only if a change of state is detected. The module modifies the network matrix to model the new state as described in Section 2.4.

3.6 Solution Module

Sixth, the program executes the Solution Module. The module uses the standard backward substitution method to compute the new nodal voltages from the LU-decomposed network matrix [16]. The module also updates the current vector [I] in the network equation (9) to be ready for the next cycle of computation.

The program repeats the execution of the Switching, Modification, and Solution Modules, until the duration of the simulation has been completed.

3.7 Harmonic Module

Lastly, the program executes the Harmonic Module. The module uses the data from the last 60Hz period in the simulation to compute the harmonics in the drive circuit. The computation method is described in Section 2.7.

4. Numerical and Field Test Validation

First, this chapter provides the numerical validation of the accuracy of the advanced PC-based program. The program is compared to the EMTP in two examples: one validates the accuracy, and the other illustrates the efficiency for modelling the switching circuits. Second, this chapter presents the results of field tests performed at LASCO (Lake Ontario Steel Company) to demonstrate the capabilities of the program. Results of measurements on one dc drive and one ac drive are used for the demonstration.

4.1 Numerical Validation

Figure 6 shows a simple circuit of an induction motor connected to a three-phase supply. Under the locked rotor condition, the motor can be modeled as a transformer and can be simulated by the PC-based EMTP. The results simulated by the EMTP and the advanced program, shown in Figure 7 are virtually the same.

4.2 Switching Simulation

Figure 8 shows the simulations of a switching operation in the circuit of Figure 6 using the EMTP and the advanced program. The numerical oscillations generated by the EMTP are overcome by the advanced program.

4.3 DC Drive Simulation

A standard dc motor drive, shown in Figure 9, consists of a 6-pulse ac-to-dc converter and a inductor choke. Figure 10 shows the input voltages of a 400-HP, 600-V dc drive that is used for straightening steel bars in the steel mill at LASCO. The results of the simulation agree reasonably well with the field measurements.

4.4 AC Drive Simulation

Figure 11 shows a standard ac motor drive which consists of an ac-to-dc converter, a dc-link filter, and a dc-to-ac inverter. This inverter is a conventional 6-step voltage source inverter (VSI), which requires a large capacitor in the dc-link filter. Figure 12 shows the output currents of a 125-HP, 600-V ac drive that is used for rotating steel bars in a steel mill. The results of the simulation agree closely with the field measurements.

5. Conclusions

The advanced power quality program for variable speed drives has overcome the difficulties encountered in existing programs, particularly:

a. The program can simulate accurately the harmonics produced by the drives, both characteristic and non-characteristic. Frequency-domain programs cannot predict the non-characteristic harmonics.

b. The program implements an automatic formulation algorithm for drive analysis. This algorithm, which requires straightforward data such as nodes and values of components, can formulate a network equation for an arbitrarily configured drive circuit. This eliminates the requirement of some time-domain programs to model a drive circuit using additional artificial resistors. This also eliminates the resulting numerical problems.

c. The program implements an efficient nodal computational algorithm for drive simulations. This substantially reduces the computation time required for the conventional state space method.

d. The program has modules for standard drives. This reduces the extensive effort needed to assemble the correct data for a power quality study.

References

[1] Cyme International Inc., "CYMHARMO - User's Guide & Reference", Nov. 1987.

[2] Cooper Industries, "V-HARM - Users Manual of Power System Harmonics Simulation and Analysis Program", May 1988.

[3] G.T. Heydt, W.M. Grady, D. Xia, "Harmonic Power Flow Studies, Vol. 1: Theoretical Basis", EPRI EL-3300, Project 1764-7, Electric Power Research Institute, Nov. 1983.

[4] J.P. Tamby, V.I. John, "Qharm - A Harmonic Powerflow Program for Small Power Systems", IEEE-PES, Summer Meeting, SM 506-9, 1987.

[5] W. Xu, J.R. Marti, H.W. Dommel, "A Multiphase Harmonic Load Flow Solution Technique", IEEE-PES WM098-4 PWRS, 1990.

[6] R.W.Y. Cheung, G.K. Hatanaka, "Advanced PC-based Program for Power Quality Analysis of Variable Speed Drives", First International Conference on Power Quality, Paris France, October 1991.

[7] R.W.Y. Cheung, J.D. Lavers, "A Basis-Transformer State Space Formulation for the Analysis of Controlled Rectifier under Ideal and Non-Ideal Steady State Conditions", IEEE Trans. Power System, PWRS-1, No. 4, 1986.

[8] R.W.Y. Cheung, J.D. Lavers, "A Basis-Transformed State Space Formulation for the Computer-Aided Design of Power Electronic Circuits and Systems", IEEE IAS Annual Meeting, Converter Analysis & Modelling Session, 1987.

[9] J.D. Lavers, H. Jin, R.W.Y. Cheung, "Analysis of Power Electronic Circuits with Feedback Control: A General Approach", IEE Proceedings, Vol. 137, Pt. B, No. 4, 1990.

[10] H.W. Dommel, "Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks", IEEE Trans. Power Apparatus & Systems, vol.PAS-88, No.4, Apr. 1969.

[11] H.W. Dommel, "Transients Program User's Manual", University of British Columbia, 1978.

[12] G.N. Revankar, S.A. Mahajan, "Digital Simulation for Mode Identification in Thyristor Circuits", Proc. IEE, vol.120, No.2, 1973.

[13] R.W.Y. Cheung, H. Jin, B. Wu, J.D. Lavers, "A Generalized Computer-Aided Formulation for the Dynamic and Steady State Analysis of Induction Machine Inverter Drive Systems", IEEE Trans. Energy Conversion, Vol. 5, No. 2, June 1990.

[14] N.N. Hancock, "Matrix Analysis of Electrical Machinery", Pergamon Press, 1974.

[15] G.E. Forsythe, M.A. Malcolm, C.B. Moler, "Computer Methods for Mathematical Computations" (Book), Prentice Hall Inc., 1977.

[16] W.F. Tinney, J.W. Walker, "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization", Proc. IEEE, Vol.55, Nov. 1967.

[17] D.A. Calahan, "Computer-Aided Network Design" (Book), McGraw-Hill, 1968.

[18] J.R. Marti, J. Lin, "Suppression of Numerical Oscillation in the EMTP", IEEE-PES Summer Meeting, 1988.

[19] T.A. Lipo, "The Analysis of Induction Motors with Voltage Control by Symmetrically Triggered Thyristors", IEEE Trans. PAS, Vol. PAS-90, No. 2, March 1971.

[20] J.D. Lavers, R.W.Y. Cheung, "A Software Package for the Steady State and Dynamic Simulation of Induction Motor Drives", IEEE Trans. Power System, PWRS-1, No. 2, May 1986.

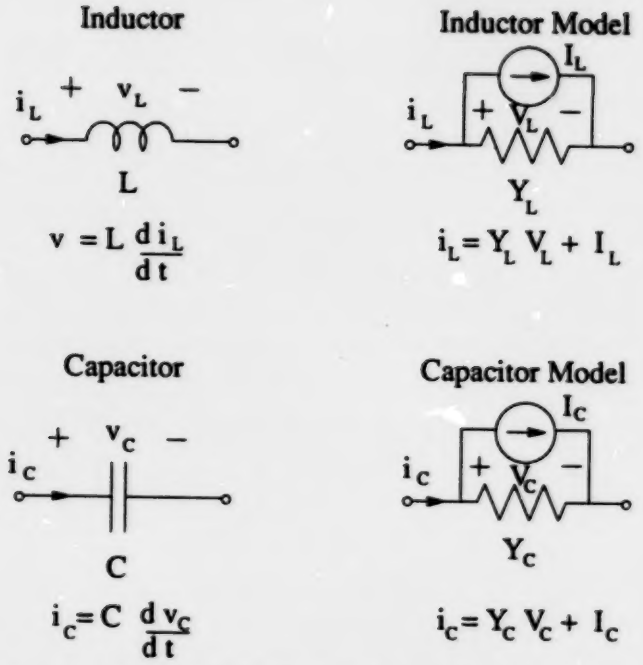


Figure 1 Nodal Formulation of Inductor and Capacitor

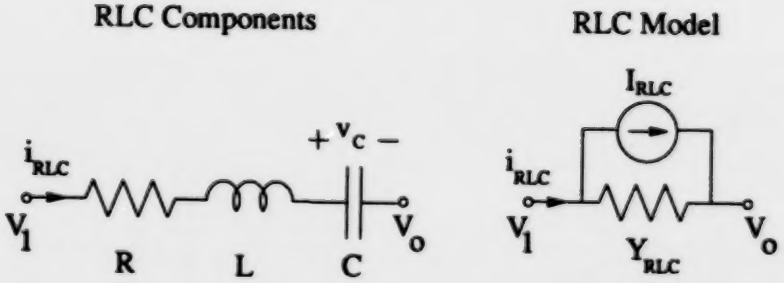


Figure 2 Nodal Formulation of a Series Connection of RLC Components

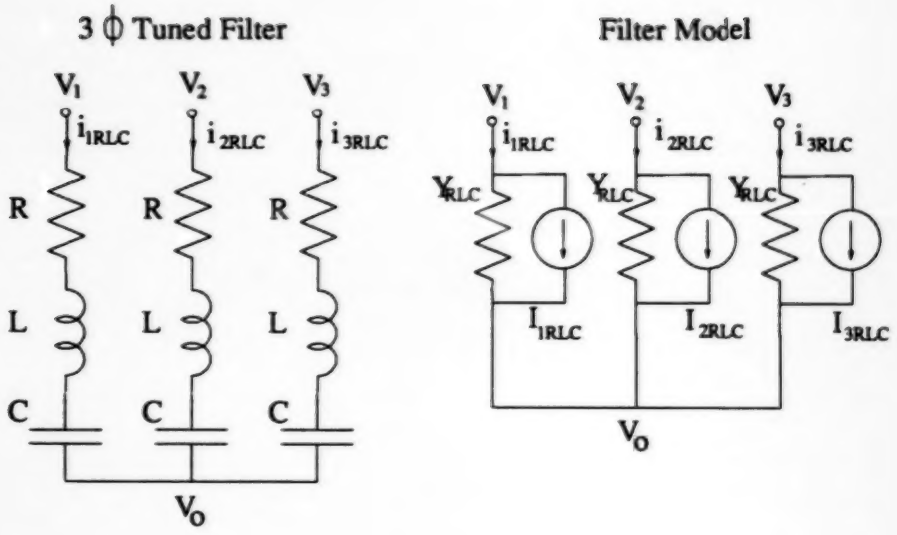
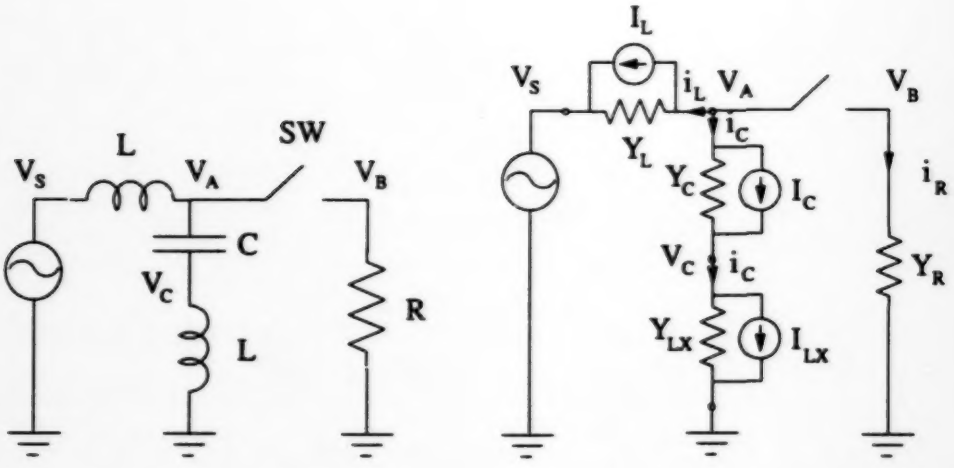


Figure 3 Nodal Formulation of a Three-Phase Tuned Filter



(a) A Switching Circuit

(b) A Nodal Equivalent Circuit

Figure 4 A Simple Circuit to Illustrate the Nodal Equation

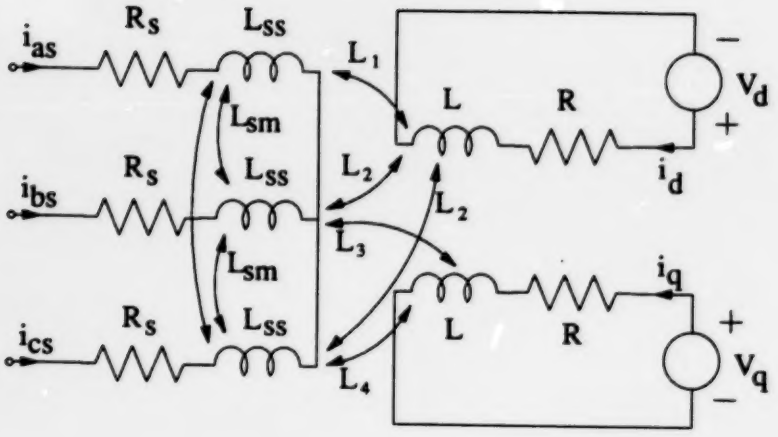


Figure 5 A Generalized Circuit Model for Induction Motor

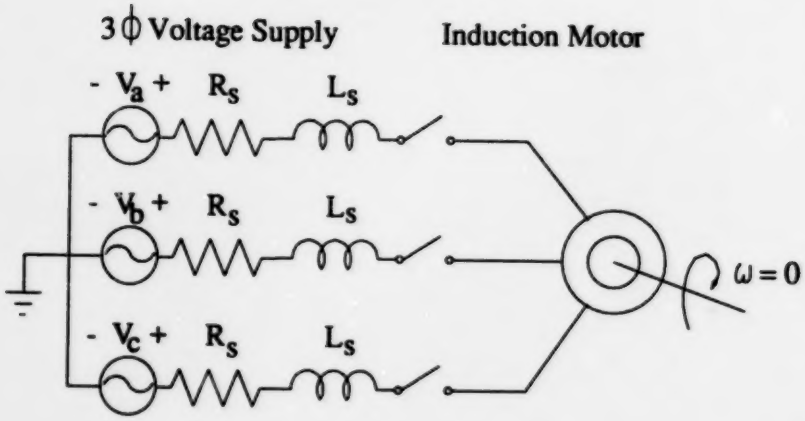
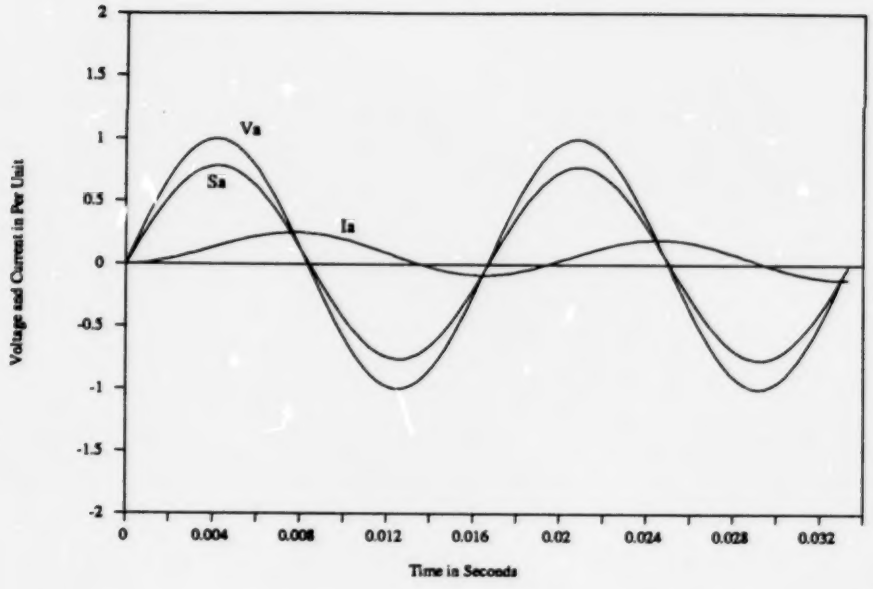
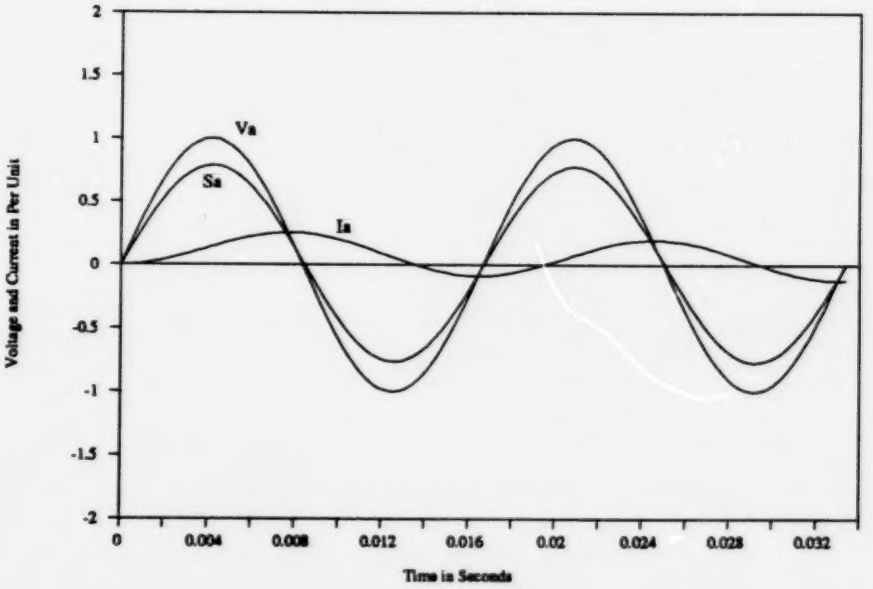


Figure 6 A Simple Circuit for Numerical Validation



ex1emtp

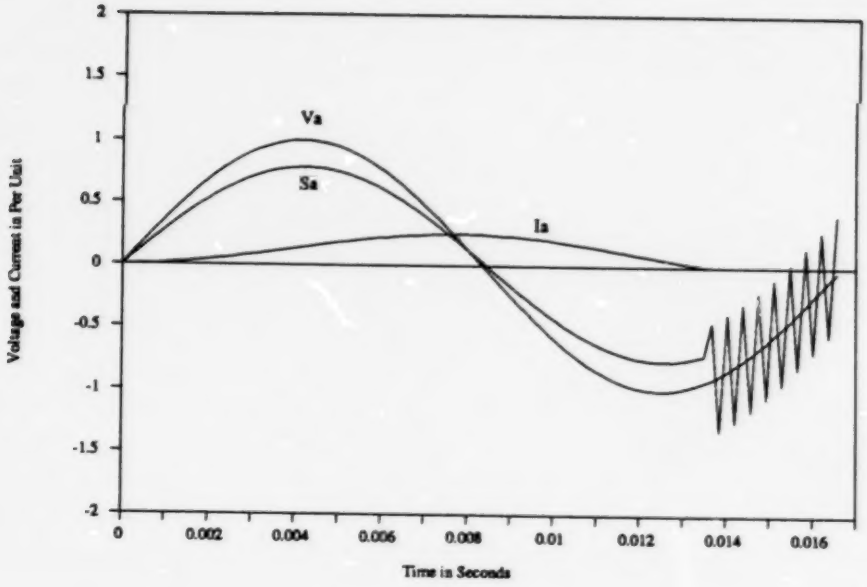
(a) EMTP Simulation



ex1pq

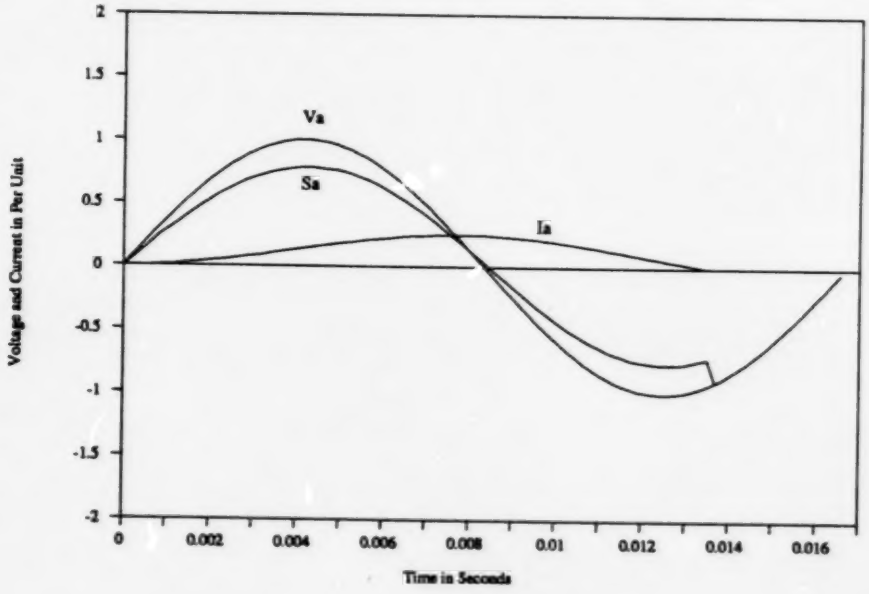
(b) Advanced PC-based Program

Figure 7 Numerical Validation



ex2emtp

(a) EMTP Simulation



ex1pq

(a) Advanced PC-based Program

Figure 8 Switching Simulation

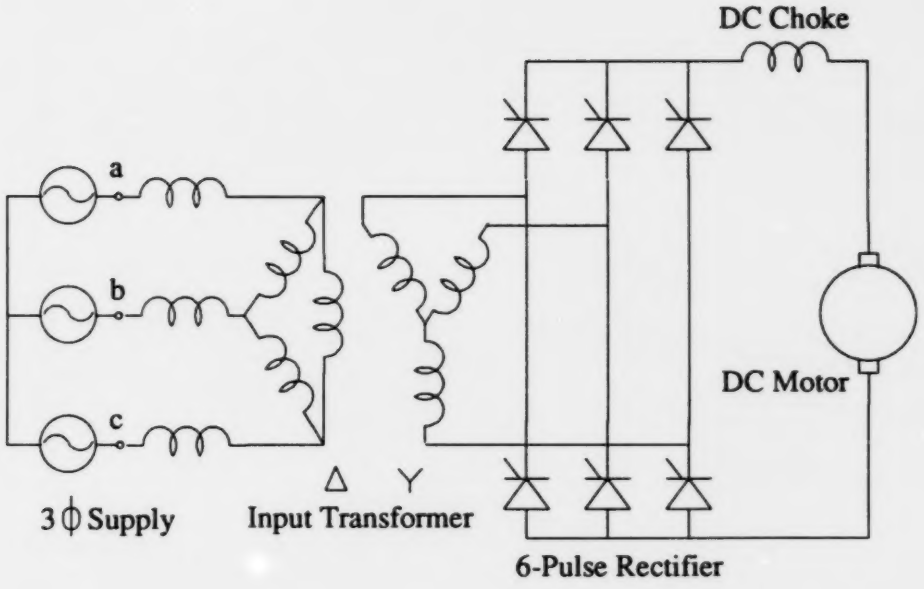


Figure 9 A DC Motor Drive

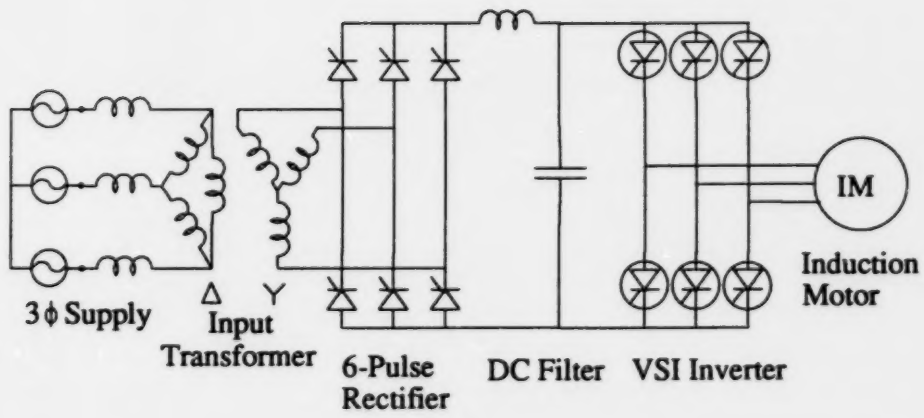
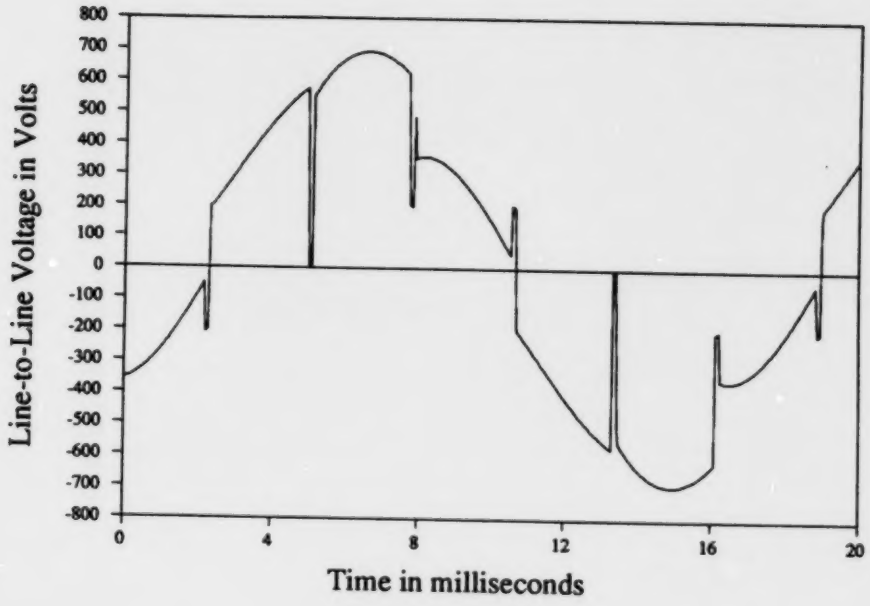
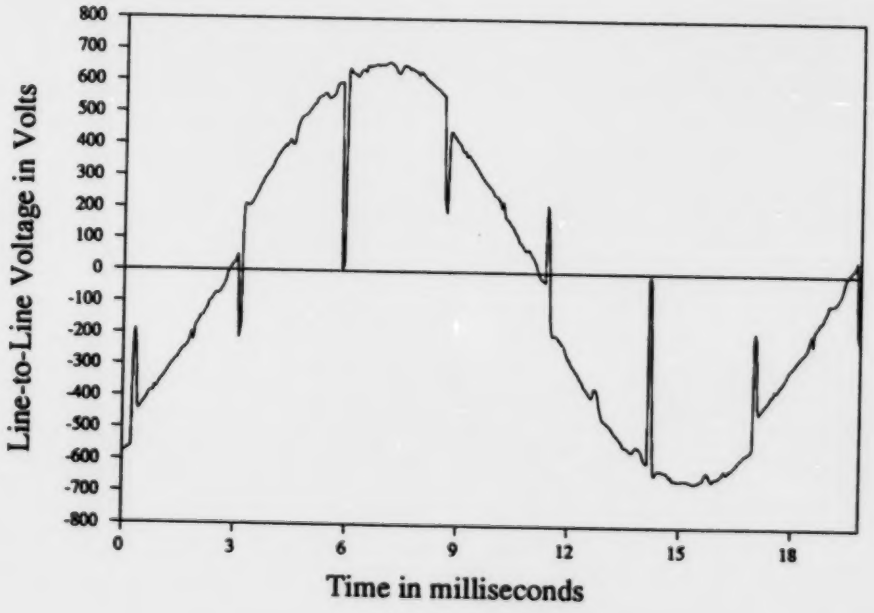


Figure 11 A Voltage-Source-Inverter Drive

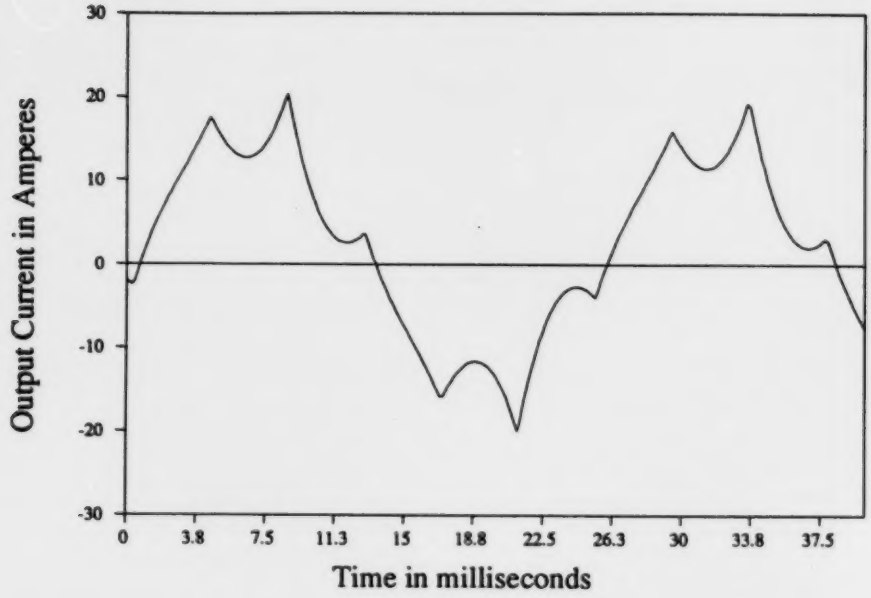


(a) Simulation

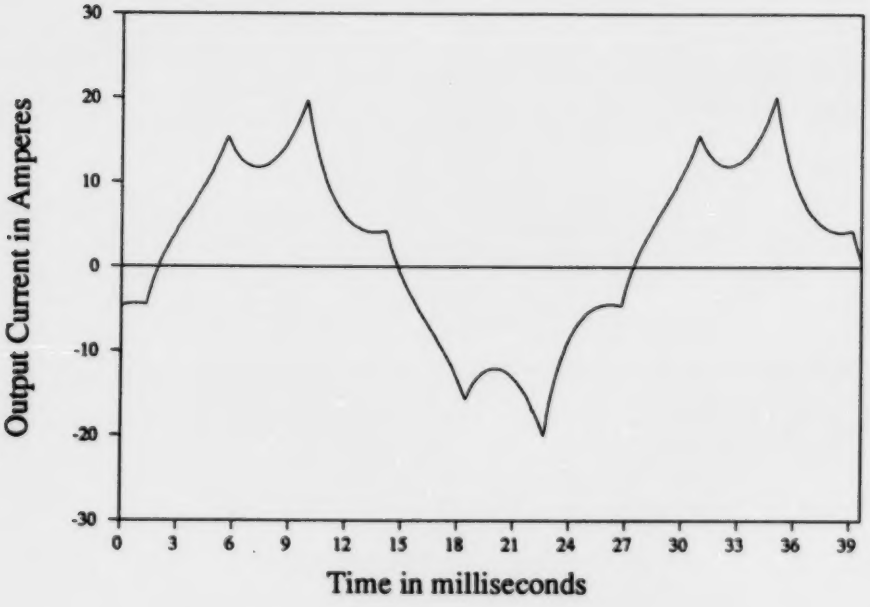


(b) Measurement

Figure 10 Simulated and Measured Input Voltages to DC Motor Drive



(a) Simulation



(b) Measurement

Figure 12 Simulated and Measured Output Currents for VSI Drive

FORTRAN SOURCE CODE FOR

ADVANCED PC-BASED PROGRAM FOR

POWER QUALITY ANALYSIS OF VARIABLE SPEED DRIVES

91-294

```
C                ************************
C                *  MAIN    PROGRAM  *
C                ************************
C
      PROGRAM   MAIN
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*12 LUNIT,LU1
      CHARACTER NAME(12),NAMDOT
c  store node names
      CHARACTER*8 CBT(20),BK,BM,BI(20),BJ(20),BT(50),BF(15)
c  store node name numbers
      DIMENSION KSWT(15),MSWT(15),KBT(30),MBT(30),KBTM(9),MBTM(9)
c  store currents
      DIMENSION SI(20),CURO(20,3),CUROM(9,2)
      DIMENSION CISS(15)
c  store voltages
      DIMENSION VV(20),VK(10,5),VN(20),SOUR(10,3)
c  store Y matrices for power circuit
      DIMENSION YL(20,20),YS(10,20),YSL(10,20),EY(30,4),TEY(9,9)
c  store Y matrices for control circuit
      DIMENSION Y(20,20),YI(20,20),YT(20,20),YCC(20,20)
c  store transfer-fun matrices
      DIMENSION DD(10,10),EE(10,10),XO(10,10)
c  store gating matrices
      DIMENSION FIRE(15,10),GAT(10,3),IFIR(15)
c  store output matrices
      DIMENSION OUT(20),IOUT(20,4)
c  store switch-checking matrices
      DIMENSION NSWT(20),NCO(15,6),ICO(15),ICOX(15),MCO(15)
      DIMENSION NOC(15,6),NOCX(6),KOC(15,6),MOC(15,6)
c  store control-cct matrices
      DIMENSION NBS(10,3),LBM(5),NBP(10,8),NBM(5),NCF(15)
c  temporary storages
      DIMENSION F(10),G(10),GUL(5,3),GG(20),GGL(20)
      DIMENSION NP(20),NODV(10),NVD(5)
c  motor
      DIMENSION MOUT(6)
c  store Fourier outputs
      DIMENSION FOURI(300,5)
      COMMON /FO1/ NFO,F1,FMAX,LCF,TF1,NFM
      COMMON /BL1/ IVD,IBV,IBS,NIBM,IBC,INC,NV,NL,NLP1,NLS,NLS1
     1,NLSV,INCX,IBMT,ICHECK,NC1,NC2,NC3
      COMMON /LB2/ T,DELT,TSTUDY,DELT2
      COMMON /LB3/ MAXC,NTC,IBVC,LIM,IBSC,IBLC,IBLM,IBLMV,ILIM,IBPV
     1,IBPC,IBCC
      COMMON /LB4/ NLSV1,ICF,NC4,NC5,NCC,IBL
      COMMON /BLS/ X1,X2,X3,P2,HH,XR,WR,TMA,TMB,TMC,MOTOR,XM
C
      EQUIVALENCE (LUNIT,NAME(1))
c
 5912 FORMAT(A)
      NC4=0
      NC5=1
      NCC=0
      ICF=0
C
C     *****  DATA INPUT & LU DECOMPOSITION FOR POWER CIRCUITS *****
```

```
C
      LU1='VSDN.TMP'
      OPEN(UNIT=6,FILE=LU1,STATUS='OLD')
      READ(6,5912) LUNIT
      read(6,*) delt,tstudy,tskip
      tskip=tskip*delt
      CLOSE(6)
    NAMDOT='.'
    DO 15 I=1,9
  15 IF(NAME(I).EQ.NAMDOT) GOTO 17
  17 I1=I+1
    NAME(I1)='o'
       I2=I1+1
       NAME(I2)='u'
       I3=I2+1
       NAME(I3)='t'
    OPEN(UNIT=4,FILE=LUNIT,STATUS='NEW')
      NAME(I1)='h'
         NAME(I2)='a'
         NAME(I3)='r'
    OPEN(UNIT=3,FILE=LUNIT,STATUS='NEW')
    CALL DATA2(INDEX,EY,TEY,GG,YL,SOUR,VN,FIRE,CURO,CUROM,IOUT,IFIR
   1,KSWT,MSWT,KBT,MBT,KBTM,MBTM,NBM,ICO,NODV,NVD,ICOX,BT,BI,BJ,BF
   2,MOUT,LU1)
    IF(INDEX.NE.2) CALL DECOMP(EY,TEY,GG,YL,YS,KBT,MBT,KBTM,MBTM,NBM
   1,NODV,NVD)
    IF(INDEX.EQ.1) GOTO 70
cc    CALL CDATA2(CBT,BI,BJ,BT,BF,Y,VV,F,G,GUL,DD,EE,XO,VK,GAT,NBS
cc   1,NBP,KBT,MBT)
cc    CALL COMP(CBT,BI,BJ,Y,YI,YT,VV,F,GUL,YCC,NBS,NP,LBM,NBP,IOUT)
  70 NC6=NC5+NCC
    T=0
    MFO=-1
    NFO=0
C
C   *****    TRANSIENT COMPUTATION    *****
C
       OPEN(UNIT=5,FILE='TEMP.DAT',STATUS='NEW')
       TCHECK=TSTUDY/10
       IJT=1
    ttskip=0
 1000 CONTINUE
       IF(T.LT.TCHECK) GOTO 1005
       WRITE(5,*) IJT
       IJT=IJT+1
       TCHECK=TSTUDY*IJT/10
 1005    CONTINUE
C
C   *****  CONTROL SIGNAL COMPUTATION  *****
C
cc    IF(INDEX.EQ.1) GOTO 45
cc    CALL CSOLVE(Y,YI,YT,VV,F,G,GUL,DD,EE,YCC,XO,VK,GAT,CURO
cc   1,OUT,NBS,LBM,NBP,NCF,IOUT)
C
C   *****  POWER CIRCUIT CURRENT COMPUTATION  *****
C
cc  45 IF(INDEX.EQ.2) GOTO 46
```

```
      CALL CURENT(EY,TEY,GG,CISS,GGL,SOUR,VN,SI,CURO,CUROM,OUT
     1,IOUT,KBT,MBT,KBTM,MBTM,NBM,NODV,NVD,NSWT,MCO,NOC,KOC,MOC,NOCX)
cc   46 continue
      if(i.lt.ttskip) goto 6708
      ttskip=ttskip+tskip
      OUT(1)=T
      IF(ICHECK.NE.2.) WRITE(4,7005) (OUT(I),I=1,NC6)
 6708 IF(T.GE.TSTUDY) GOTO 2000
      IF(INDEX.EQ.2) GOTO 10
C
C     *****  MONITORING CHANGES OF STATES IN POWER CIRCUIT  *****
C
      IF(ICHECK.EQ.3) ICHECK=0
      IF(ICHECK.NE.0) GOTO 10
      IF(IBS.EQ.0) GOTO 10
      CALL SWITCH(GG,YS,GGL,VN,YSL,CISS,FIRE,KSWT,MSWT,NSWT
     1,NCO,ICO,MCO,NOC,KOC,MOC,ICOX,IFIR,SI,CURO,CUROM
     2,NCF)
      IF(ICHECK.EQ.1)
     6 CALL CURENT(EY,TEY,GG,CISS,GGL,SOUR,VN,SI,CURO,CUROM,OUT
     1,IOUT,KBT,MBT,KBTM,MBTM,NBM,NODV,NVD,NSWT,MCO,NOC,KOC,MOC,NOCX)
    10 IF(LCF.EQ.0) GOTO 30
      IF(T.LT.TF1) GOTO 30
      MFO=MFO+1
      IF(MFO.LT.NFM) GOTO 30
      NFO=NFO+1
      DO 24 J=1,LCF
      K=IOUT(J,4)
    24 FOURI(NFO,J)=OUT(K)
      MFO=-1
    30 T=T+DELT
      IF(INDEX.EQ.2) GOTO 1000
C
C     *****  POWER CIRCUIT SOLUTION  *****
C
      IF(ICHECK.EQ.0) GOTO 35
      T=T-DELT/2.
      ICHECK=ICHECK+1
    35 CALL SOLVE(MCO,NCO,NODV,NVD,YL,GGL,SOUR,VN,YSL,SI,OUT,IOUT
     1,CURO,MOUT)
C
      GOTO 1000
 2000 CONTINUE
C
C     *****  FORURIER ANALYSIS  *****
C
      IF(LCF.NE.0) CALL PQ(FOURI)
 7001 FORMAT(I5)
 7005 FORMAT(10E12.4)
      CLOSE(3)
      CLOSE(4)
      CLOSE(5)
      CLOSE(6)
      STOP
      END
C                  **********************
C                  *  Fourier Analysis  *
```

```
C                    **********************
C
      SUBROUTINE PQ(FOURI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION FOURI(300,5),FMAG(5),FANG(5)
      COMMON /FO1/ NFO,F1,FMAX,LCF,TF1,NFM
      PI2NFO=6.283185/NFO
      S1=2./NFO
      MAX=FMAX/F1+1
      FREQ=0.
      DO 20 M=1,LCF
      DC=0.
      DO 10 J=1,NFO
   10 DC=DC+FOURI(J,M)
      FMAG(M)=DC/NFO
   20 FANG(M)=0.
      WRITE(3,6123) FREQ,(FMAG(I),FANG(I),I=1,LCF)
      DO 100 L=2,MAX
      K=L-1
      DO 70 M=1,LCF
      A=0.
      B=0.
      DO 50 J=1,NFO
      S=PI2NFO*(J-1)*K
      A=A+FOURI(J,M)*DCOS(S)
   50 B=B+FOURI(J,M)*DSIN(S)
      FMAG(M)=S1*DSQRT(A*A+B*B)
   70 FANG(M)=57.2958*DATAN2(A,B)
      FREQ=K*F1
      WRITE(3,6123) FREQ,(FMAG(I),FANG(I),I=1,LCF)
  100 CONTINUE
 6123 FORMAT(1X,11E12.4)
      RETURN
      END
C                    ****************************
C                    *  POWER-CCT DATA INPUT    *
C                    ****************************
C
      SUBROUTINE DATA2(INDEX,EY,TEY,GG,YL,SOUR,VN,FIRE,CURO,CUROM
     1,IOUT,IFIR,KSWT,MSWT,KBT,MBT,KBTM,MBTM,NBM,ICO,NODV,NVD,ICOX,BT
     2,BI,BJ,BF,MOUT,LU1)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*70 DUMP
      CHARACTER*12 LU1
      CHARACTER MOST(8),  STRIN(100),HI,QE,STRI(100)
      CHARACTER*2 MOT2
      CHARACTER*8 MOTT
      CHARACTER*8 BT(50),BK,BM,BI(20),BJ(20),BF(15),BLANK,MOTA,MOTB,MOTC
      CHARACTER*8 PA,PB,PC,PN,SA,SB,SC,SN,MOTP,MOTN,MOTPE,MOTNE
      DIMENSION EY(30,4),TEY(9,9),GG(20),SOUR(10,3),VN(20),FIRE(15,10)
      DIMENSION CURO(20,3),CUROM(9,2),OUT(20),YL(20,20),NP(12)
      DIMENSION KSWT(15),MSWT(15),KBT(30),MBT(30),KBTM(9),MBTM(9),NBM(5)
      DIMENSION ICO(15),NODV(10),NVD(5),ICOX(15),IFIR(15),IOUT(20,4)
      DIMENSION MOUT(6)
      COMMON /BL1/ IVD,IBV,IBS,NIBM,IBC,INC,NV,NL,NLP1,NLS,NLS1
     1,NLSV,INCX,IBMT,ICHECK,NC1,NC2,NC3
      COMMON /LB2/ T,DELT,TSTUDY,DELT2
```

```
      COMMON /LB4/ NLSV1,ICF,NC4,NC5,NCC,IBL
      COMMON /BLS/ X1,X2,X3,P2,HH,XR,WR,TMA,TMB,TMC,MOTOR,XM
      COMMON /PO1/ NPO,F1,FMAX,LCF,TF1,NFM
      COMMON /PM1/ XTS,XTM,NM,NT,MAX,LC,ERR,EYX,ECC
      COMMON /MO1/ MOTA,MOTB,MOTC,RS,RR
      EQUIVALENCE (MOTT,MOST(1)),(MOST(7),MOT2)
      MAX=50
      BT(MAX)='      '
      NT=MAX
      BLANK ='      '
C-----------------------------------------------------------
      BK='TEMP.DAT'
      OPEN(UNIT=5,FILE=BK,STATUS='NEW')
      DO 291 J=1,8
      CLOSE(6)
      OPEN(UNIT=6,FILE=LU1,STATUS='OLD')
cc
       READ(6,5009) DUMP
cc
      READ(6,*) DELT,TSTUDY
        TSTUDY=TSTUDY/60.
  292 READ(6,5008) JTYPE,DUMP
 5008 FORMAT(I5,A70)
      IF(JTYPE.EQ.0) GOTO 291
      IF(JTYPE.NE.J) GOTO 292
      WRITE(5,5009) DUMP
 5009 FORMAT(A70)
      GOTO 292
  291 WRITE(5,6293) JTYPE
 6293 FORMAT(20X,I5)
      CLOSE(5)
      OPEN(UNIT=5,FILE=BK,STATUS='OLD')
C-----------------------------------------------------------
C      READ(5,*) INDEX
       INDEX=1
      IF(INDEX.EQ.2) RETURN
      MOTOR=0
C 3 Terminal Names of Induction Motor
      READ(5,5006)  MOTA,MOTB,MOTC
      IF(MOTA.EQ.MOTB) GOTO 12
cc      write(*,6301)
cc 6301 format(' ac motor')
      MOTOR=1
C Motor Parameters: RS,RR,XS,XM,XR
      READ(5,*)    RS,RR,XS,XM,XR
C Motor Parameters: P2,HH,TMA,TMB,TMC
      READ(5,*)    P2,HH,TMA,TMB,TMC
      READ(5,*) (MOUT(I),I=1,6)
      WR=0.
      XS=XS*1.E-3
      XM=XM*1.E-3
      XR=XR*1.E-3
   12 IVD=0
      IF(MOTOR.EQ.1) READ(5,*) ITYPE
C DC Motor
      MOTORD=0
      READ(5,5006) MOTP,MOTN
```

```
      IF(MOTP.EQ.MOTN) GOTO 100
cc      write(*,6302)
cc 6302 format(' dc motor')
      MOTORD=1
      READ(5,*) EDC,RDC,XDC
      READ(5,*) IDC
      MOTPE='MOTp'
      MOTNE='MOTn'
      NT=NT-1
      BT(NT)=MOTPE
      IVD=IVD+1
      BT(IVD)=MOTNE
      NODV(IVD)=NT
      SOUR(IVD,1)=EDC
      SOUR(IVD,2)=0.
      SOUR(IVD,3)=0.
C-----------------------------------------------------------------
  100 CONTINUE
      IF(MOTORD.EQ.1) READ(5,*) ITYPE
C Type of Floating Voltage Source
      READ(5,*)  ITYPE
      IF(ITYPE.EQ.0) GOTO 109
cc      write(*,6303)
cc 6303 format(' floating voltage')
C 2 Terminal Names
      READ(5,5006) BK,BM
C Voltage, Frequency, Phase
      READ(5,*)    VM,FV,PV
         READ(5,*) IJK
      NT=NT-1
      BT(NT)=BK
      IVD=IVD+1
      BT(IVD)=BM
      NODV(IVD)=NT
      SOUR(IVD,1)=VM
      SOUR(IVD,2)=FV*6.28319
      SOUR(IVD,3)=PV*0.0174533
      GOTO 100
  109 IBV=IVD
C-----------------------------------------------------------------
  110 CONTINUE
C Type of Grounded Voltage source
      READ(5,*) ITYPE
      IF(ITYPE.EQ.0) GOTO 119
cc      write(*,6304)
cc 6304 format(' grounded voltage')
C Terminal Name(s)
      READ(5,5006) BI(1),BI(2),BI(3)
C Voltage, Fequency, Phase
      READ(5,*)    VM,FV,PV
         READ(5,*) IJK
      DO 503 I=1,ITYPE
      NT=NT-1
      BT(NT)=BI(I)
      IBV=IBV+1
      NODV(IBV)=NT
      SOUR(IBV,1)=VM
```

```
      SOUR(IBV,2)=FV*6.28319
      SOUR(IBV,3)=PV*0.0174533
  503 PV=PV-360/ITYPE
      GOTO 110
  119 IF(MOTOR.EQ.0) GOTO 14
      NT=NT-1
      MOTT=MOTA
      MOT2='vd'
      BT(NT)=MOTT
      IBV=IBV+1
      NODV(IBV)=NT
      KBT(4)=NT
      NT=NT-1
      MOT2='vq'
      BT(NT)=MOTT
      IBV=IBV+1
      NODV(IBV)=NT
      KBT(5)=NT
   14 NV=IBV+1
      LC=0
      IBS=0
      ICF=0
      LCF=0
C-----------------------------------------------------------------------
  120 CONTINUE
C Type of Switch
      READ(5,*) ITYPE
      IF(ITYPE.EQ.0) GOTO 129
cc      write(*,6305)
cc 6305 format(' switch')
      IF(ITYPE.NE.6) GOTO 126
      CALL SIXCON(IBS,NT,MAX,BT,FIRE,IFIR,KSWT,MSWT)
      GOTO 1261
C 2 Terminal Name(s)
  126 READ(5,5006) BK,BM
      IBS=IBS+1
      IF(ITYPE.GT.2) GOTO 127
C Number, Frequency, Switchings <time/angle>
      READ(5,*) N,FV, (FIRE(IBS,J),J=1,N)
      FIRE(IBS,N+1)=TSTUDY*2
      IF(FV.EQ.0) GOTO 127
      FV=1./FV
      DO 125 J=1,N
  125 FIRE(IBS,J)=FIRE(IBS,J)*FV/360.
      FIRE(IBS,N+1)=FV
      IF(ITYPE.EQ.2) N=-N
      IFIR(IBS)=N
      GOTO 128
  127 IF(ITYPE.EQ.1) K=N+1
      IF(ITYPE.EQ.2) K=-N-1
      IF(ITYPE.EQ.3) K=100
      IF(ITYPE.EQ.4) K=-100
      IF(ITYPE.EQ.5) K=0
      IFIR(IBS)=K
      IF(IABS(K).NE.100) GOTO 128
      ICF=ICF+1
C Name of Close-Loop-Control Switch
```

```
      READ(5,5006) BF(ICF)
  128 CONTINUE
      DO 121 K=NT,MAX
  121 IF(BK.EQ.BT(K)) GOTO 122
      NT=NT-1
      BT(NT)=BK
      K=NT
  122 KSWT(IBS)=K
      DO 123 M=NT,MAX
  123 IF(BM.EQ.BT(M)) GOTO 124
      NT=NT-1
      BT(NT)=BM
      M=NT
  124 MSWT(IBS)=M
c     IF(IOC.EQ.0) GOTO 120
c     LC=LC+1
c     IOUT(LC,1)=IBS
c     BI(LC)=BK
c     BJ(LC)=BM
 1261   READ(5,*) IJK
      GOTO 120
  129 NS=MAX-NT+1-NV
      NC1=LC
C
      EYDELT=2.E-3/DELT
      ECDELT=2.E-6/DELT
      NM=0
      NIBM=0
      IF(MOTOR.NE.0)
     1 CALL IMOT1(NIBM,NBM,MBT,KBTM,MBTM,BT,XS,DELT,YL,TEY,MOTA)
C-----------------------------------------------------------------------
  158 CONTINUE
C Mutual Couplings
      READ(5,*) ITYPE
      IF(ITYPE.EQ.0) GOTO 159
cc      write(*,6306)
cc 6306 format(' coupling')
      IF(ITYPE.EQ.5) GOTO 157
      READ(5,*) XTS,XTM
      XTS=(XTS+XTM)*EYDELT
      XTM=XTM*EYDELT
      IF(ITYPE.NE.1) GOTO 151
      READ(5,5006) PA,PN,SA,SN
      READ(5,*) IAP
      CALL TRANS1(PA,PN,SA,SN,IAP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      GOTO 158
  151 READ(5,5006) PA,PB,PC,PN,SA,SB,SC,SN
      READ(5,*) IAP,IBP,ICP
      IF(ITYPE.NE.2) GOTO 152
      CALL TRANS1(PA,PN,SA,SN,IAP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      CALL TRANS1(PB,PN,SB,SN,IBP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      CALL TRANS1(PC,PN,SC,SN,ICP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      GOTO 158
```

```
  152 IF(ITYPE.NE.3) GOTO 153
      CALL TRANS1(PA,PN,SA,SB,IAP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      CALL TRANS1(PB,PN,SB,SC,IBP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      CALL TRANS1(PC,PN,SC,SA,ICP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      GOTO 158
  153 CALL TRANS1(PA,PB,SA,SB,IAP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      CALL TRANS1(PB,PC,SB,SC,IBP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      CALL TRANS1(PC,PA,SC,SA,ICP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      GOTO 158
  157 READ(5,*) IBM
      NIBM=NIBM+1
      NBM(NIBM)=IBM
      DO 140 IB=1,IBM
C Two Names of Coupling Branch
      READ(5,5006) BK,BM
C Current Output, One row of Inductances
      READ(5,*)    IOC,(GG(J),J=1,IB)
      N=NM+IB
      DO 141 K=NT,MAX
  141 IF(BK.EQ.BT(K)) GOTO 142
      NT=NT-1
      BT(NT)=BK
      K=NT
  142 KBTM(N)=K
      DO 143 M=NT,MAX
  143 IF(BM.EQ.BT(M)) GOTO 144
      NT=NT-1
      BT(NT)=BM
      M=NT
  144 MBTM(N)=M
      DO 145 J=1,IB
  145 YL(IB,J)=-GG(J)*EYDELT
      IF(IOC.EQ.0) GOTO 140
      LC=LC+1
      IOUT(LC,1)=N
      BI(LC)=BK
      BJ(LC)=BM
      IF(IOC.EQ.1) GOTO 140
      LCF=LCF+1
      IOUT(LCF,4)=LC+1
  140 CONTINUE
      CALL TRANSY(IBM,NM,YL,TEY)
      GOTO 158
  159 NC2=LC
C
      IBMT=NM
      IBL=0
      IF(MOTOR.NE.0) CALL IMOT2(IBL,MOUT,BT,BI,BJ,KBT,MBT,JOUT,EY)
      IF(MOTORD.EQ.0) GOTO 130
      ERR=RDC
      EYX=0.
```

```
      ECC=0.
      CALL RLC(MOTP,MOTPE,IDC,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      ERR=0.
      EYX=XDC*EYDELT
      IDC=0
      CALL RLC(MOTN,MOTNE,IDC,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
C-----------------------------------------------------------------------
  130 CONTINUE
C Type of RLC Branch
      READ(5,*) ITYPE
      IF(ITYPE.EQ.0) GOTO 139
cc      write(*,6307)
cc 6307 format(' rlc')
C Terminal Names of RLC Branch
      READ(5,5006) PA,PB,PC,SA,SB,SC
C Resistance, Inductance, Capacitance
      READ(5,*) ERR,EYX,ECC
      ECC=ECC*ECDELT
      IF(ECC.NE.0) ECC=1./ECC
      EYX=EYX*EYDELT
      READ(5,*) IAP,IBP,ICP
      IF(ITYPE.NE.1) GOTO 131
      CALL RLC(PA,PB,IAP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      GOTO 130
  131 IF(ITYPE.NE.2) GOTO 132
      CALL PIRLC(PB,SB,IBP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      GOTO 130
  132 IF(ITYPE.NE.3) GOTO 133
      CALL RLC(PA,SA,IAP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      CALL RLC(PB,SB,IBP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      CALL RLC(PC,SC,ICP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      GOTO 130
  133 CALL PIRLC(PA,SA,IAP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      CALL PIRLC(PB,SB,IBP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      CALL PIRLC(PC,SC,ICP,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      GOTO 130
  139 NC3=LC
      IF(IVD.EQ.0) GOTO 170
      DO 165 I=1,IVD
      BM=BT(I)
      DO 166 M=NT,MAX
  166 IF(BM.EQ.BT(M)) GOTO 165
      NT=NT-1
      BT(NT)=BM
      M=NT
  165 NVD(I)=M
  170 NL=MAX-NT+1-NV-NS
      NT1=NT-1
      IBC=0
C-----------------------------------------------------------------------
  160 CONTINUE
C Type of Current Source
      READ(5,*) ITYPE
      IF(ITYPE.EQ.0) GOTO 169
cc      write(*,6308)
cc 6308 format(' current source')
C Terminal Names
```

```
      READ(5,5006) BK,BM
C Current, Frequency, Phase
      READ(5,*)    CM,FV,PV
  163 IBC=IBC+1
      J=IBC+IBV
      DO 161 K=NT,MAX
  161 IF(BK.EQ.BT(K)) GOTO 162
      WRITE(*,6002)
 6002 FORMAT(1X,'ERROR CURRENT INPUT')
  162 NODV(J)=K-NT1
      SOUR(J,1)=CM
      SOUR(J,2)=FV*6.28319
      SOUR(J,3)=PV*0.0174533
      IF(BM.EQ.BT(MAX)) GOTO 160
      BK=BM
      BM=BT(MAX)
      CM=-CM
      GOTO 163
  169 CONTINUE
      DO 172 I=NT,MAX
  172 BT(I-NT1)=BT(I)
      IF(IBS.EQ.0) GOTO 173
      DO 174 I=1,IBS
      KSWT(I)=KSWT(I)-NT1
  174 MSWT(I)=MSWT(I)-NT1
  173 IF(IBL.EQ.0) GOTO 175
      DO 176 I=1,IBL
      KBT(I)=KBT(I)-NT1
  176 MBT(I)=MBT(I)-NT1
  175 IF(NIBM.EQ.0) GOTO 177
      DO 178 I=1,NM
      KBTM(I)=KBTM(I)-NT1
  178 MBTM(I)=MBTM(I)-NT1
  177 IF(IVD.EQ.0) GOTO 179
      DO 181 I=1,IVD
  181 NVD(I)=NVD(I)-NT1
  179 CONTINUE
      DO 182 I=1,IBV
  182 NODV(I)=NODV(I)-NT1
      NLP1=NL+1
      NLS=NL+NS
      NLS1=NLS+1
      NLSV=NLS+NV-1
      NLSV1=NLSV+1
      DO 183 K=1,NLSV1
  183 VN(K)=0
C-------------------------------------------------
      IF(IBL.EQ.0) GOTO 191
      DO 195 I=1,IBL
      CURO(I,1)=0.
      CURO(I,2)=0.
  195 CURO(I,3)=0.
  191 IF(NIBM.EQ.0) GOTO 196
      NM=0
      DO 201 IB=1,NIBM
      IBM=NBM(IB)
  201 NM=NM+IBM
```

```
      DO 203 N=1,NM
      CUROM(N,1)=0.
  203 CUROM(N,2)=0.
  196 CONTINUE
C-------------------------------------------------------------------
C Node Names for Voltage Outputs
cc    READ(6,*) NC4
cc    IF(NC4.EQ.0) GOTO 230
      NC31=NC3+1
cc    K=NC3+NC4
        K=NC3+5
        NC4=0
      READ(6,5006) (BI(I),BJ(I),I=NC31,K)
        DO 250 I=NC31,K
        IF(BI(I).EQ.BJ(I)) GOTO 251
  250   NC4=NC4+1
  251   IF(NC4.EQ.0) GOTO 230
      READ(6,*) (NP(I),I=1,NC4)
cc    write(*,6309)
cc 6309 format(' volt out')
      DO 220 I=1,NC4
      BK=BI(I+NC3)
      BM=BJ(I+NC3)
      DO 212 K=1,NLSV1
  212 IF(BK.EQ.BT(K)) GOTO 214
      WRITE(*,6006)
 6006 FORMAT(' ERROR')
      STOP
  214 IOUT(I,2)=K
      DO 216 M=1,NLSV1
  216 IF(BM.EQ.BT(M)) GOTO 218
      WRITE(*,6006)
      STOP
  218 IOUT(I,3)=M
      IF(NP(I).EQ.0) GOTO 220
      LCF=LCF+1
      IOUT(LCF,4)=I+NC3+1
  220 CONTINUE
  230 NC5=NC3+NC4+1
      IF(MOTOR.EQ.0) GOTO 566
      TORE=0.
      TORM=TMA+WR*(TMB+TMC*WR)
      IF(MOUT(4).EQ.0) GOTO 5661
      BI(NC5)='SPEED'
      NC5=NC5+1
      OUT(NC5)=WR
 5661 IF(MOUT(5).EQ.0) GOTO 5662
      BI(NC5)='E.TOR'
      NC5=NC5+1
      OUT(NC5)=TORE
 5662 IF(MOUT(6).EQ.0) GOTO 566
      BI(NC5)='M.TOR'
      NC5=NC5+1
      OUT(NC5)=TORM
  566 CONTINUE
      K=NC5-1
      CLOSE(5)
```

```
      OPEN(UNIT=5,FILE='TEMP.DAT',STATUS='NEW')
      BK='gnd'
      DO 590 I=1,K
      IF(BI(I).EQ.BLANK) BI(I)=BK
  590 IF(BJ(I).EQ.BLANK) BJ(I)=BK
      HI='-'
      QE='?'
      BK='?I:'
      BM='?V:'
      IF(K.GT.0) GOTO 564
      WRITE(*,6564)
 6564 FORMAT(' Error: No Output !')
      STOP
  564 IF(NC3.EQ.0) GOTO 562
      IF(K.GE.NC31) WRITE(5,5007)
     1(BK,BI(I),HI,BJ(I),I=1,NC3),(BM,BI(J),HI,BJ(J),J=NC31,K)
      IF(K.LT.NC31) WRITE(5,5007) (BK,BI(I),HI,BJ(I),I=1,NC3)
      GOTO 563
  562 WRITE(5,5007) (BM,BI(J),HI,BJ(J),J=NC31,K)
  563 CONTINUE
 5007 FORMAT(20(A3,A6,A1,A6))
      CLOSE(5)
      OPEN(UNIT=5,FILE='TEMP.DAT',STATUS='OLD')
      READ(5,50071) (STRIN(I),I=1,100)
50071 FORMAT(100A1)
      N=0
      HI=' '
      DO 565 I=1,100
      IF(STRIN(I).EQ.HI) GOTO 565
      IF(STRIN(I).EQ.QE) STRIN(I)=HI
      N=N+1
      STRI(N)=STRIN(I)
  565 CONTINUE
      BK='Time'
      BM='Freq'
      WRITE(4,50072) BK,(STRI(I),I=1,N)
      WRITE(3,50072) BM,(STRI(I),I=1,N)
50072 FORMAT(A4,100A1)
C
      IF(IBS.EQ.0) GOTO 561
      DO 560 L=1,IBS
      ICOX(L)=0
  560 ICO(L)=0
  561 INCX=0
      ICHECK=0
      IF(LCF.EQ.0) GOTO 570
      NFM=300
      M=0
C-------------------------------------------------------------
C Fourier Analysis: F1, Fmax
      READ(6,*)     F1,FMAX
      TF1=1./F1
      N=TF1/DELT
      TF1=TSTUDY-TF1
      IF(TF1.GT.0.) GOTO 572
      WRITE(*,7125)
 7125  FORMAT(' ERROR: NOT A PERIOD')
```

```
      STOP
  572 IF(N.LE.NFM) GOTO 571
      N=N/2
      M=M+1
      GOTO 572
  571 NFM=M
  570 CONTINUE
 5006 FORMAT(12A6)
      RETURN
      END
C
c                              Pi-RLC circuit
c
      SUBROUTINE PIRLC(BK,BM,IOC,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*8 BT(50),BK,BM,BI(20),BJ(20)
      DIMENSION EY(30,4),KBT(30),MBT(30),IOUT(20,4)
      COMMON /PM1/ XTS,XTM,NM,NT,MAX,LC,ERR,EYX,ECC
      IBL=IBL+1
      DO 131 K=NT,MAX
  131 IF(BK.EQ.BT(K)) GOTO 132
      NT=NT-1
      BT(NT)=BK
      K=NT
  132 KBT(IBL)=K
      DO 133 M=NT,MAX
  133 IF(BM.EQ.BT(M)) GOTO 134
      NT=NT-1
      BT(NT)=BM
      M=NT
  134 MBT(IBL)=M
      TEMP=1./(ERR+EYX)
      EY(IBL,1)=TEMP
      EY(IBL,2)=(ERR-EYX)*TEMP
      EY(IBL,3)=0.
      EY(IBL,4)=EYX
      IF(IOC.EQ.0) GOTO 135
      LC=LC+1
      IOUT(LC,1)=IBL
      BI(LC)=BK
      BJ(LC)=BM
      IF(IOC.EQ.1) GOTO 135
      LCF=LCF+1
      IOUT(LCF,4)=LC+1
  135 IBL=IBL+1
      KBT(IBL)=K
      MBT(IBL)=MAX
      EY(IBL,1)=1./ECC
      EY(IBL,2)=1.
      EY(IBL,3)=ECC
      EY(IBL,4)=0.
      IBL=IBL+1
      KBT(IBL)=M
      MBT(IBL)=MAX
      EY(IBL,1)=1./ECC
      EY(IBL,2)=1.
      EY(IBL,3)=ECC
```

```
      EY(IBL,4)=0.
      RETURN
      END
C
c                                   Series RLC circuit
c
      SUBROUTINE RLC(BK,BM,IOC,KBT,MBT,BT,BI,BJ,EY,IOUT,IBL,LCF)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*8 BT(50),BK,BM,BI(20),BJ(20)
      DIMENSION EY(30,4),KBT(30),MBT(30),IOUT(20,4)
      COMMON /PM1/ XTS,XTM,NM,NT,MAX,LC,ERR,EYX,ECC
      IBL=IBL+1
      DO 131 K=NT,MAX
  131 IF(BK.EQ.BT(K)) GOTO 132
      NT=NT-1
      BT(NT)=BK
      K=NT
  132 KBT(IBL)=K
      DO 133 M=NT,MAX
  133 IF(BM.EQ.BT(M)) GOTO 134
      NT=NT-1
      BT(NT)=BM
      M=NT
  134 MBT(IBL)=M
      TEMP=1./(ERR+EYX+ECC)
      EY(IBL,1)=TEMP
      EY(IBL,2)=(ERR-EYX+ECC)*TEMP
      EY(IBL,3)=ECC
      EY(IBL,4)=EYX
      IF(IOC.EQ.0) RETURN
      LC=LC+1
      IOUT(LC,1)=IBL
      BI(LC)=BK
      BJ(LC)=BM
      IF(IOC.EQ.1) RETURN
      LCF=LCF+1
      IOUT(LCF,4)=LC+1
      RETURN
      END
C
c                              Resistive part of Induction motor
c
      SUBROUTINE IMOT2(IBL,MOUT,BT,BI,BJ,KBT,MBT,IOUT,EY)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*8 BT(50),BI(20),BJ(20),MOTA,MOTB,MOTC
      DIMENSION KBT(30),MBT(30),IOUT(20,4),EY(30,4),MOUT(6)
      COMMON /PM1/ XTS,XTM,NM,NT,MAX,LC,ERR,EYX,ECC
      COMMON /MO1/ MOTA,MOTB,MOTC,RS,RR
      IBL=5
      DO 22 K=NT,MAX
   22 IF(MOTA.EQ.BT(K)) GOTO 23
      NT=NT-1
      BT(NT)=MOTA
      K=NT
   23 KBT(1)=K
      DO 24 K=NT,MAX
   24 IF(MOTB.EQ.BT(K)) GOTO 25
```

```
      NT=NT-1
      BT(NT)=MOTB
      K=NT
   25 KBT(2)=K
      DO 26 K=NT,MAX
   26 IF(MOTC.EQ.BT(K)) GOTO 27
      NT=NT-1
      BT(NT)=MOTC
      K=NT
   27 KBT(3)=K
      S=1./RS
      EY(1,1)=S
      EY(2,1)=S
      EY(3,1)=S
      S=1./RR
      EY(4,1)=S
      EY(5,1)=S
      DO 28 I=1,5
      EY(I,2)=1.
      EY(I,4)=0.
   28 EY(I,3)=0.
      DO 30 I=1,3
      IF(MOUT(I).EQ.0) GOTO 30
      LC=LC+1
      IOUT(LC,1)=I
      BI(LC)=BT(KBT(I))
      BJ(LC)=BT(MBT(I))
   30 CONTINUE
      RETURN
      END
C
c                    Magnetic coupling part of induction motor
c
      SUBROUTINE IMOT1(NIBM,NBM,MBT,KBTM,MBTM,BT,XS,DELT,YL,TEY,MOTA)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER MO8T(8)
      CHARACTER*2 MOT2
      CHARACTER*8 BT(50),MOTT,MOTA
      DIMENSION YL(20,20),TEY(9,9)
      DIMENSION MBT(30),KBTM(9),MBTM(9),NBM(5)
      COMMON /BL5/ X1,X2,X3,P2,HH,XR,WR,TMA,TMB,TMC,MOTOR,XM
      COMMON /PM1/ XTS,XTM,NM,NT,MAX,LC,ERR,EYX,ECC
      EQUIVALENCE (MOTT,MO8T(1)),(MO8T(7),MOT2)
      NIBM=1
      IBM=5
      NBM(1)=5
      MOTT=MOTA
      MOT2='-a'
      BT(NT-1)=MOTT
      MOT2='-b'
      BT(NT-2)=MOTT
      MOT2='-c'
      BT(NT-3)=MOTT
      MOT2='-d'
      BT(NT-4)=MOTT
      MOT2='-q'
      BT(NT-5)=MOTT
```

```
      MOT2='-n'
      BT(NT-6)=MOTT
      DO 16 I=1,5
      NT=NT-1
      KBTM(I)=NT
   16 MBT(I)=NT
      NT=NT-1
      MBTM(1)=NT
      MBTM(2)=NT
      MBTM(3)=NT
      MBTM(4)=MAX
      MBTM(5)=MAX
      S=-XS*2./DELT
      YL(1,1)=S
      YL(2,2)=S
      YL(3,3)=S
      YL(2,1)=0.
      YL(3,1)=0.
      YL(3,2)=0.
      S=-XR*2./DELT
      YL(4,4)=S
      YL(5,5)=S
      X1=0.816497*XM
      YL(4,1)=-X1*2./DELT
      X2=-0.408248*XM
      YL(4,2)=-X2*2./DELT
      YL(4,3)=YL(4,2)
      X3=0.707107*XM
      YL(5,2)=-X3*2./DELT
      YL(5,3)=-YL(5,2)
      YL(5,1)=0.
      YL(5,4)=0.
      CALL TRANSY(IBM,NM,YL,TEY)
      RETURN
      END
C
c                              1-phase transformer
c
      SUBROUTINE TRANS1(PA,PN,SA,SN,IOCP,NIBM,BT,BI,BJ,YL,TEY
     1,NBM,KBTM,MBTM,IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*8 PA,PN,SA,SN,BT(50),BI(20),BJ(20)
      DIMENSION KBTM(9),MBTM(9),IOUT(20,4),NBM(5)
      DIMENSION YL(20,20),TEY(9,9)
      COMMON /PM1/ XTS,XTM,NM,NT,MAX,LC,ERR,EYX,ECC
      IBM=2
      NIBM=NIBM+1
      NBM(NIBM)=IBM
      N=NM+1
      DO 141 K=NT,MAX
  141 IF(PA.EQ.BT(K)) GOTO 142
      NT=NT-1
      BT(NT)=PA
      K=NT
  142 KBTM(N)=K
      DO 143 M=NT,MAX
  143 IF(PN.EQ.BT(M)) GOTO 144
```

```
      NT=NT-1
      BT(NT)=PN
      M=NT
  144 MBTM(N)=M
      IF(IOCP.EQ.0) GOTO 140
      LC=LC+1
      IOUT(LC,1)=N
      BI(LC)=PA
      BJ(LC)=PN
      IF(IOCP.EQ.1) GOTO 140
      LCF=LCF+1
      IOUT(LCF,4)=LC+1
  140 N=NM+2
      DO 145 K=NT,MAX
  145 IF(SA.EQ.BT(K)) GOTO 146
      NT=NT-1
      BT(NT)=SA
      K=NT
  146 KBTM(N)=K
      DO 147 M=NT,MAX
  147 IF(SN.EQ.BT(M)) GOTO 148
      NT=NT-1
      BT(NT)=SN
      M=NT
  148 MBTM(N)=M
      YL(1,1)=-XTS
      YL(2,2)=-XTS
      YL(2,1)=-XTM
      CALL TRANSY(IBM,NM,YL,TEY)
      RETURN
      END
C
c                          General magnetic coupling circuit
c
      SUBROUTINE TRANSY(IBM,NM,YL,TEY)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION YL(20,20),TEY(9,9)
      DO 147 I=1,IBM
      DO 147 J=I,IBM
  147 YL(I,J)=YL(J,I)
      DO 148 N=1,IBM
      IF(N.EQ.1) GOTO 153
      DO 149 J=1,IBM
      TEMP=YL(1,J)
      YL(1,J)=YL(N,J)
  149 YL(N,J)=TEMP
      DO 150 I=1,IBM
      TEMP=YL(I,1)
      YL(I,1)=YL(I,N)
  150 YL(I,N)=TEMP
  153 TEMP=-1./YL(1,1)
      YL(1,1)=TEMP
      DO 151 J=2,IBM
  151 YL(1,J)=YL(1,J)*TEMP
      DO 152 I=2,IBM
      TEMP=YL(I,1)
      DO 152 J=I,IBM
```

```
152 YL(I,J)=YL(I,J)+YL(1,J)*TEMP
    DO 154 J=1,IBM
    DO 154 I=J,IBM
154 YL(I,J)=YL(J,I)
    IF(N.EQ.1) GOTO 148
    DO 155 J=1,IBM
    TEMP=YL(1,J)
    YL(1,J)=YL(N,J)
155 YL(N,J)=TEMP
    DO 156 I=1,IBM
    TEMP=YL(I,1)
    YL(I,1)=YL(I,N)
156 YL(I,N)=TEMP
148 CONTINUE
    DO 146 I=1,IBM
    K=I+NM
    DO 146 J=1,IBM
146 TEY(K,J)=YL(I,J)
    NM=NM+IBM
    RETURN
    END
C
c                          6-pulse bridge
c
    SUBROUTINE SIXCON(IBS,NT,MAX,BT,FIRE,IFIR,KSWT,MSWT)
    IMPLICIT REAL*8 (A-H,O-Z)
    CHARACTER*8 PA,PB,PC,PP,PN,BT(50)
    DIMENSION FIRE(15,10),IFIR(15),KSWT(15),MSWT(15)
    READ(5,5006) PA,PB,PC,PP,PN
    READ(5,*) ALPHA,FV
    FV=1./FV
    ALPHA=ALPHA*FV/360.
    T60D=FV/6.
    T120D=T60D*2.
    IBS=IBS+1
    NTX=NT
    DO 21 K=NTX,MAX
21  IF(PA.EQ.BT(K)) GOTO 22
    NT=NT-1
    BT(NT)=PA
    K=NT
22  KSWT(IBS)=K
    MSWT(IBS+3)=K
    DO 31 K=NTX,MAX
31  IF(PB.EQ.BT(K)) GOTO 32
    NT=NT-1
    BT(NT)=PB
    K=NT
32  KSWT(IBS+2)=K
    MSWT(IBS+5)=K
    DO 41 K=NTX,MAX
41  IF(PC.EQ.BT(K)) GOTO 42
    NT=NT-1
    BT(NT)=PC
    K=NT
42  KSWT(IBS+4)=K
    MSWT(IBS+1)=K
```

```
      DO 51 K=NTX,MAX
   51 IF(PN.EQ.BT(K)) GOTO 52
      NT=NT-1
      BT(NT)=PN
      K=NT
   52 KSWT(IBS+1)=K
      KSWT(IBS+3)=K
      KSWT(IBS+5)=K
      DO 61 K=NTX,MAX
   61 IF(PP.EQ.BT(K)) GOTO 62
      NT=NT-1
      BT(NT)=PP
      K=NT
   62 MSWT(IBS)=K
      MSWT(IBS+2)=K
      MSWT(IBS+4)=K
      DO 100 I=1,6
      BETA=ALPHA+T120D
      FIRE(IBS,1)=ALPHA-FV
      FIRE(IBS,2)=BETA-FV
      FIRE(IBS,3)=ALPHA
      FIRE(IBS,4)=BETA
      FIRE(IBS,5)=FV
      IFIR(IBS)=4
      IBS=IBS+1
  100 ALPHA=ALPHA+T60D
 5006 FORMAT(12A6)
      RETURN
      END
C                     **********************
C                     * LU DECOMPOSITION *
C                     **********************
C
      SUBROUTINE DECOMP(EY,TEY,GG,YL,YS,KBT,MBT,KBTM,MBTM,NBM,NODV,NVD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION EY(30,4),TEY(9,9),GG(20),YL(20,20),YS(10,20)
      DIMENSION KBT(30),MBT(30),KBTM(9),MBTM(9),NODV(10),NVD(5),NBM(5)
      COMMON /BL1/ IVD,IBV,IBS,NIBM,IBC,INC,NV,NL,NLP1,NLS,NLS1
     1,NLSV,INCX,IBMT,ICHECK,NC1,NC2,NC3
      COMMON /LB2/ T,DELT,TSTUDY,DELT2
      COMMON /LB4/ NLSV1,ICF,NC4,NC5,NCC,IBL
C-------------------------------------------------------------
C       SYSTEM MATRIX FORMATION
C-------------------------------------------------------------
      DO 500 L=1,NLS
      I=NLSV+1
      DO 504 J=1,I
  504 GG(J)=0
      LY=0
      LS=L
  550 CONTINUE
      IF(IBL.EQ.0) GOTO 509
      DO 506 I=1,IBL
      K=KBT(I)
      M=MBT(I)
      IF(K.EQ.LS) GOTO 508
      IF(M.EQ.LS) GOTO 507
```

```
      GOTO 506
  507 M=K
  508 TEMP=EY(I,1)
      GG(LS)=GG(LS)+TEMP
      GG(M)=GG(M)-TEMP
  506 CONTINUE
  509 IF(NIBM.EQ.0) GOTO 519
      NM=0
      DO 511 IB=1,NIBM
      IBM=NBM(IB)
      DO 510 I=1,IBM
      N=NM+I
      IF(KBTM(N).EQ.LS) GOTO 512
      IF(MBTM(N).EQ.LS) GOTO 516
      GOTO 510
  512 DO 514 J=1,IBM
      NJ=NM+J
      K=KBTM(NJ)
      M=MBTM(NJ)
      TEMP=TEY(N,J)
      GG(K)=GG(K)+TEMP
  514 GG(M)=GG(M)-TEMP
      GOTO 510
  516 DO 518 J=1,IBM
      NJ=NM+J
      K=KBTM(NJ)
      M=MBTM(NJ)
      TEMP=TEY(N,J)
      GG(M)=GG(M)+TEMP
  518 GG(K)=GG(K)-TEMP
  510 CONTINUE
  511 NM=NM+IBM
  519 IF(LY.EQ.IVD) GOTO 552
      LY=LY+1
      M=NVD(LY)
      IF(M.NE.L) GOTO 519
      LS=NODV(LY)
      GOTO 550
  552 IF(IVD.EQ.0) GOTO 520
      DO 554 I=1,IVD
      K=NODV(I)
      M=NVD(I)
  554 GG(M)=GG(M)+GG(K)
  520 CONTINUE
C-----------------------------------------------------
C       DECOMPOSITION
C-----------------------------------------------------
      IF(L.EQ.1) GOTO 526
      IF(L.GT.NL) GOTO 530
      K=L-1
      DO 522 I=1,K
      TEMP=GG(I)
      M=I+1
      DO 524 J=M,NLSV
  524 GG(J)=GG(J)-TEMP*YL(I,J)
  522 CONTINUE
  526 TEMP=1./GG(L)
```

```
      YL(L,L)=1EMP
      K=L+1
      DO 528 J=K,NLSV
  528 YL(L,J)=GG(J)*TEMP
      GOTO 500
  530 DO 532 I=1,NL
      TEMP=GG(I)
      M=I+1
      DO 534 J=M,NLSV
  534 GG(J)=GG(J)-TEMP*YL(I,J)
  532 CONTINUE
      K=L-NL
      DO 536 J=NLP1,NLSV
  536 YS(K,J)=GG(J)
  500 CONTINUE
      RETURN
      END
C                     ************************
C                     * CURRENT COMPUTATION *
C                     ************************
C
      SUBROUTINE CURENT(EY,TEY,GG,CISS,GGL,SOUR,VN,SI,CURO,CUROM,OUT
     1,IOUT,KBT,MBT,KBTM,MBTM,NBM,NODV,NVD,NSWT,MCO,NOC,KOC,MOC,NOCX)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION EY(30,4),TEY(9,9),GG(20),CISS(15)
      DIMENSION GGL(20),SOUR(10,3),VN(20),SI(20)
      DIMENSION CURO(20,3),CUROM(9,2),OUT(20)
      DIMENSION KBT(30),MBT(30),KBTM(9),MBTM(9),NODV(10),NVD(5)
      DIMENSION NSWT(20),MCO(15),NOC(15,6),KOC(15,6),MOC(15,6),NOCX(6)
      DIMENSION IOUT(20,4),NBM(5)
      COMMON /BL1/ IVD,IBV,IBS,NIBM,IBC,INC,NV,NL,NLP1,NLS,NLS1
     1,NLSV,INCX,IBMT,ICHECK,NC1,NC2,NC3
      COMMON /LB2/ T,DELT,TSTUDY,DELT2
      COMMON /LB4/ NLSV1,ICF,NC4,NC5,NCC,IBL
C-----------------------------------------------------------------
C       COMPUTE BRANCH CURRENTS
C-----------------------------------------------------------------
      DO 802 I=1,NLSV1
  802 SI(I)=0
      IF(IBL.EQ.0) GOTO 809
      DO 804 L=1,IBL
      K=KBT(L)
      M=MBT(L)
      YEV=EY(L,1)*(VN(K)-VN(M))
      CI1=YEV+CURO(L,2)
      IF(ICHECK.EQ.0) GOTO 803
      CV1=CURO(L,3)+EY(L,3)*CI1
      EI1=EY(L,1)*( EY(L,4)*CI1-CV1 )
      IF(ICHECK.EQ.1) CV1=CURO(L,3)+EY(L,3)*(CI1+CURO(L,1))
      IF(ICHECK.EQ.3) EI1=-EY(L,2)*CI1+YEV-2.*EY(L,1)*CV1
      GOTO 805
  803 CV1=CURO(L,3)+EY(L,3)*(CI1+CURO(L,1))
      YEV=YEV-2.*EY(L,1)*CV1
      EI1=-EY(L,2)*CI1+YEV
  805 SI(K)=SI(K)-EI1
      SI(M)=SI(M)+EI1
      CURO(L,1)=CI1
```

```
      CURO(L,2)=EI1
      CURO(L,3)=CV1
  804 CONTINUE
  809 IF(NIBM.EQ.0) GOTO 859
      NM=0
      DO 850 IB=1,NIBM
      IBM=NBM(IB)
      DO 852 L=1,IBM
      N=NM+L
      K=KBTM(N)
      M=MBTM(N)
  852 GG(L)=VN(K)-VN(M)
      DO 854 I=1,IBM
      N=NM+I
      GGL(I)=0
      DO 854 J=1,IBM
  854 GGL(I)=GGL(I)+TEY(N,J)*GG(J)
      DO 856 I=1,IBM
      N=NM+I
      CUROM(N,1)=GGL(I)+CUROM(N,2)
      CUROM(N,2)=CUROM(N,1)
  856 IF(ICHECK.EQ.0) CUROM(N,2)=GGL(I)+CUROM(N,1)
      DO 858 L=1,IBM
      N=NM+L
      K=KBTM(N)
      M=MBTM(N)
      SI(K)=SI(K)-CUROM(N,2)
  858 SI(M)=SI(M)+CUROM(N,2)
  850 NM=NM+IBM
  859 IF(IVD.EQ.0) GOTO 806
      DO 807 I=1,IVD
      K=NODV(I)
      M=NVD(I)
  807 SI(M)=SI(M)+SI(K)
  806 CONTINUE
          IF(ICHECK.EQ.1) RETURN
C-----------------------------------------------------------
C       COMPUTE SWITCH CURRENTS
C-----------------------------------------------------------
      IF(IBS.EQ.0) GOTO 839
      DO 826 I=1,IBS
  826 CISS(I)=0.
      IF(INCX.EQ.0) GOTO 839
      DO 820 I=NLP1,NLS
      GG(I)=0
      N=NSWT(I)
      IF(N.EQ.0) GOTO 820
      IF(IBL.EQ.0) GOTO 821
      DO 822 L=1,IBL
      IF(KBT(L).EQ.I) GG(I)=GG(I)-CURO(L,1)
  822 IF(MBT(L).EQ.I) GG(I)=GG(I)+CURO(L,1)
  821 IF(NIBM.EQ.0) GOTO 825
      NM=0
      DO 827 IB=1,NIBM
      IBM=NBM(IB)
      DO 823 L=1,IBM
      N=NM+L
```

```
      IF(KBTM(N).EQ.I) GG(I)=GG(I)-CUROM(N,1)
  823 IF(MBTM(N).EQ.I) GG(I)=GG(I)+CUROM(N,1)
  827 NM=NM+IBM
  825 CONTINUE
      IF(IBC.EQ.0) GOTO 820
      DO 824 L=1,IBC
      J=L+IBV
      TEMP=SOUR(J,1)*DCOS(SOUR(J,2)*T+SOUR(J,3))
  824 IF(NODV(J).EQ.I) GG(I)=GG(I)+TEMP
  820 CONTINUE
      DO 830 L=1,INCX
      M=MCO(L)
      IF(M.LT.0) M=-M
      M1=M-1
      DO 831 I=1,M1
  831 NOCX(I)=NOC(L,I)
  840 CONTINUE
      DO 832 I=1,M1
      IF(NOCX(I).EQ.0) GOTO 832
      K=KOC(L,I)
      M=MOC(L,I)
      IF(K.GT.NLS) GOTO 836
      DO 834 J=1,M1
      IF(I.EQ.J) GOTO 834
      IF(NOCX(J).EQ.0) GOTO 834
      IF(K.EQ.KOC(L,J)) GOTO 836
      IF(K.EQ.MOC(L,J)) GOTO 836
  834 CONTINUE
      N=NOCX(I)
      CISS(N)=GG(K)
      NOCX(I)=0
      GG(M)=GG(M)+GG(K)
      GOTO 840
  836 IF(M.GT.NLS) GOTO 832
      DO 838 J=1,M1
      IF(J.EQ.I) GOTO 838
      IF(NOCX(J).EQ.0) GOTO 838
      IF(M.EQ.KOC(L,J)) GOTO 832
      IF(M.EQ.MOC(L,J)) GOTO 832
  838 CONTINUE
      N=NOCX(I)
      CISS(N)=-GG(M)
      NOCX(I)=0
      GG(K)=GG(K)+GG(M)
      GOTO 840
  832 CONTINUE
  830 CONTINUE
  839 CONTINUE
C-----------------------------------------------------------------
C       STORE OUTPUT CURRENTS
C-----------------------------------------------------------------
      L=0
  602 IF(NC1.EQ.L) GOTO 604
      L=L+1
      K=IOUT(L,1)
      OUT(L+1)=CISS(K)
      GOTO 602
```

```
  604 IF(NC2.EQ.L) GOTO 606
      L=L+1
      K=IOUT(L,1)
      OUT(L+1)=CUROM(K,1)
      GOTO 604
  606 IF(NC3.EQ.L) GOTO 608
      L=L+1
      K=IOUT(L,1)
      OUT(L+1)=CURO(K,1)
      GOTO 606
  608 CONTINUE
      RETURN
      END
C                  **********************
C                  * SWITCH  STATES *
C                  **********************
C
      SUBROUTINE  SWITCH(GG,YS,GGL,VN,YSL,CISS,FIRE,KSWT,MSWT,NSWT
     1,NCO,ICO,MCO,NOC,KOC,MOC,ICOX,IFIR,SI,CURO,CUROM
     2,NCF)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION GG(20),YS(10,20),GGL(20),VN(20),YSL(10,20),FIRE(15,10)
      DIMENSION CISS(15),SI(20),CURO(20,3),CUROM(9,2)
      DIMENSION KSWT(15),MSWT(15),NSWT(20),NCO(15,6),ICO(15),MCO(15)
      DIMENSION NOC(15,6),KOC(15,6),MOC(15,6),ICOX(15),IFIR(15),NCF(15)
      COMMON /BL1/ IVD,IBV,IBS,NIBM,IBC,INC,NV,NL,NLP1,NLS,NLS1
     1,NLSV,INCX,IBMT,ICHECK,NC1,NC2,NC3
      COMMON /LB2/ T,DELT,TSTUDY,DELT2
      COMMON /LB4/ NLSV1,ICF,NC4,NC5,NCC,IBL
C-------------------------------------------------------------
C     CHECK GATINGS
C-------------------------------------------------------------
      JCF=0
      ICHECK=0
      DO 902 L=1,IBS
      N=IFIR(L)
      IF(ICOX(L).EQ.1) GOTO 904
      K=KSWT(L)
      M=MSWT(L)
      VKM=VN(K)-VN(M)
      IF(VKM.LE.0.) GOTO  902
      IF(N.EQ.0) GOTO 908
      IF(N.LT.0) N=-N
      IF(N.NE.100) GOTO 901
      JCF=JCF+1
      K=NCF(JCF)
      IF(K.GT.0) GOTO 908
      GOTO 902
  901 CONTINUE
      DO 903 I=1,N
  903 IF(T.LT.FIRE(L,I)) GOTO 905
      FV=FIRE(L,N+1)
      DO 907 I=1,N
  907 FIRE(L,I)=FIRE(L,I)+FV
      GOTO 901
  905 J=I/2
      IF(J*2.NE.I) GOTO 902
```

```
 908 ICHECK=1
     ICOX(L)=1
     GOTO 902
 904 IF(N.GE.0) GOTO 909
     N=-N
     IF(N.NE.100) GOTO 911
     JCF=JCF+1
     K=NCF(JCF)
     IF(K.EQ.0) GOTO 918
     GOTO 909
 911 CONTINUE
     DO 913 I=1,N
 913 IF(T.LT.FIRE(L,I)) GOTO 915
     FV=FIRE(L,N+1)
     DO 917 I=1,N
 917 FIRE(L,I)=FIRE(L,I)+FV
     GOTO 911
 915 J=I/2
     IF(J*2.NE.I) GOTO 918
 909 IF(CISS(L).GT.0) GOTO 902
 918 ICHECK=1
     ICOX(L)=0
 902 ICO(L)=ICOX(L)
     IF(T.EQ.0) GOTO 600
     IF(ICHECK.EQ.0) GOTO 809
C-----------------------------------------------------------------------
C       CHECK SWITCH STATES
C-----------------------------------------------------------------------
 600 CONTINUE
     DO 601 J=NLP1,NLS
 601 NSWT(J)=0
     INC=0
     DO 630 L=1,IBS
     IF(ICO(L).EQ.0) GOTO 630
     NEG=1
     K=KSWT(L)
     M=MSWT(L)
     ISW=2
     INC=INC+1
     NCO(INC,1)=K
     NCO(INC,2)=M
     NOC(INC,1)=L
     KOC(INC,1)=K
     MOC(INC,1)=M
     IF(K.GT.NLS) NEG=-K
     IF(M.GT.NLS) NEG=-M
     ICO(L)=0
     NSWT(K)=1
     NSWT(M)=1
     IF(IBS.EQ.L) GOTO 614
     L1=L+1
 612 CONTINUE
     DO 604 I=L1,IBS
     IF(ICO(I).EQ.0) GOTO 604
     K=KSWT(I)
     M=MSWT(I)
     KM=M
```

```
      DO 606 J=1,ISW
      JJ=NCO(INC,J)
      IF(K.EQ.JJ) GOTO 608
  606 IF(M.EQ.JJ) GOTO 610
      GOTO 604
  610 KM=K
  608 ISW=ISW+1
      NCO(INC,ISW)=KM
      ISW1=ISW-1
      NOC(INC,ISW1)=I
      KOC(INC,ISW1)=K
      MOC(INC,ISW1)=M
      IF(KM.GT.NLS) NEG=-KM
      ICO(I)=0
      NSWT(KM)=1
      GOTO 612
  604 CONTINUE
  614 MCO(INC)=ISW
      IF(NEG.EQ.1) GOTO 630
      MCO(INC)=-ISW
      NCO(INC,ISW+1)=-NEG
  630 CONTINUE
      INCX=INC
      DO 632 I=NLP1,NLS
      K=NSWT(I)
      IF(K.EQ.1) GOTO 632
      INC=INC+1
      NCO(INC,1)=I
      MCO(INC)=1
  632 CONTINUE
C-----------------------------------------------------------------
C       CHANGE SYSTEM EQUATIONS
C-----------------------------------------------------------------
      LS=0
      DO 640 N=1,INC
      M=MCO(N)
      IF(M.LT.0) GOTO 640
      LS=LS+1
      K=NCO(N,1)-NL
      DO 642 J=NLP1,NLSV
  642 GG(J)=YS(K,J)
      IF(M.EQ.1) GOTO 648
      DO 644 I=2,M
      K=NCO(N,I)-NL
      DO 646 J=NLP1,NLSV
  646 GG(J)=GG(J)+YS(K,J)
  644 CONTINUE
  648 LY=0
      DO 649 J=NLS1,NLSV
  649 GGL(J)=GG(J)
      DO 650 K=1,INC
      M=MCO(K)
      IF(M.LT.0) GOTO 653
      LY=LY+1
      GGL(LY)=0
      DO 652 J=1,M
      JJ=NCO(K,J)
```

```
652 GGL(LY)=GGL(LY)+GG(JJ)
    GOTO 650
653 M=-M
    JL=NCO(K,M+1)
    GGL(JL)=0
    DO 654 J=1,M
    JJ=NCO(K,J)
654 GGL(JL)=GGL(JL)+GG(JJ)
650 CONTINUE
    IF(LS.EQ.1) GOTO 656
    K=LS-1
    DO 670 I=1,K
    TEMP=GGL(I)
    M=I+1
    DO 672 J=M,LY
672 GGL(J)=GGL(J)-TEMP*YSL(I,J)
    DO 674 J=NLS1,NLSV
674 GGL(J)=GGL(J)-TEMP*YSL(I,J)
670 CONTINUE
656 TEMP=1./GGL(LS)
    YSL(LS,LS)=TEMP
    K=LS+1
    IF(K.GT.LY) GOTO 660
    DO 658 J=K,LY
658 YSL(LS,J)=GGL(J)*TEMP
660 CONTINUE
    DO 662 J=NLS1,NLSV
662 YSL(LS,J)=GGL(J)*TEMP
640 CONTINUE
    DO 828 I=1,INC
    M=MCO(I)
    IF(M.GT.0) GOTO 828
    M=-M
    K=NCO(I,M+1)
    J=0
    DO 829 JJ=1,M
    N=NCO(I,JJ)
    IF(N.EQ.K) GOTO 829
    J=J+1
    NCO(I,J)=N
829 CONTINUE
    NCO(I,M)=K
828 CONTINUE
809 CONTINUE
    RETURN
    END
C                   **********************
C                   *  LU    SOLVER  *
C                   **********************
C
    SUBROUTINE SOLVE(MCO,NCO,NODV,NVD,YL,GGL,SOUR,VN,YSL,SI,OUT
   1,IOUT,CURO,MOUT)
    IMPLICIT REAL*8 (A-H,O-Z)
    DIMENSION YL(20,20),GGL(20),SOUR(10,3),VN(20),YSL(10,20)
    DIMENSION SI(20),OUT(20),CURO(20,3)
    DIMENSION MCO(15),NCO(15,6),NODV(10),NVD(5),IOUT(20,4),MOUT(6)
    COMMON /BL1/ IVD,IBV,IBS,NIBM,IBC,INC,NV,NL,NLP1,NLS,NLS1
```

```
     1,NLSV,INCX,IBMT,ICHECK,NC1,NC2,NC3
      COMMON /LB2/ T,DELT,TSTUDY,DELT2
      COMMON /LB4/ NLSV1,ICF,NC4,NC5,NCC,IBL
      COMMON /BLS/ X1,X2,X3,P2,HH,XR,WR,TMA,TMB,TMC,MOTOR,XM
      IBV2=IBV-2
      IF(MOTOR.EQ.0) IBV2=IBV
      DO 810 I=1,IBV2
      N=NODV(I)
  810 VN(N)=SOUR(I,1)*DCOS(SOUR(I,2)*T+SOUR(I,3))
      IF(MOTOR.EQ.0) GOTO 814
      C1=CURO(1,1)
      C2=CURO(2,1)
      C3=CURO(3,1)
      C4=CURO(4,1)
      C5=CURO(5,1)
      N=NODV(IBV2+1)
      VN(N)=P2/2.*WR*(X3*(-C2+C3)-XR*C5)
      N=NODV(IBV)
      VN(N)=P2/2.*WR*(X1*C1+X2*(C2+C3)+XR*C4)
      TORE=P2/2.*XM*( (0.707107*C1+1.41421*C2)*C4-1.224745*C1*C5 )
      TORM=TMA+WR*(TMB+TMC*WR)
      WR=WR+(TORE-TORM)*DELT/HH
      NN=NC5
      IF(MOUT(6).EQ.0) GOTO 10
      OUT(NN)=TORM
      NN=NN-1
   10 IF(MOUT(5).EQ.0) GOTO 11
      OUT(NN)=TORE
      NN=NN-1
   11 IF(MOUT(4).EQ.0) GOTO 814
      OUT(NN)=WR
  814 IF(IBC.EQ.0) GOTO 811
      DO 812 I=1,IBC
      J=I+IBV
      N=NODV(J)
      IF(IVD.EQ.0) GOTO 812
      DO 815 ii=1,IVD
      K=NODV(ii)
  815 IF(K.EQ.N) N=NVD(ii)
  812 SI(N)=SI(N)+SOUR(J,1)*DCOS(SOUR(J,2)*T+SOUR(J,3))
  811 CONTINUE
C-------------------------------------------------------------
C       FORWARD  COMPUTATION
C-------------------------------------------------------------
      DO 702 I=1,NL
      TEMP=SI(I)
      SI(I)=YL(I,I)*TEMP
      K=I+1
      DO 702 J=K,NLS
  702 SI(J)=SI(J)-YL(I,J)*TEMP
      IF(IBS.EQ.0) GOTO 740
      LS=0
      DO 704 N=1,INC
      M=MCO(N)
      IF(M.LT.0) GOTO 704
      LS=LS+1
      GGL(LS)=0
```

```
      DO 705 J=1,M
      JJ=NCO(N,J)
  705 GGL(LS)=GGL(LS)+SI(JJ)
  704 CONTINUE
      IF(LS.EQ.0) GOTO 799
      DO 706 I=1,LS
      TEMP=GGL(I)
      GGL(I)=YSL(I,I)*TEMP
      K=I+1
      IF(K.GT.LS) GOTO 707
      DO 706 J=K,LS
  706 GGL(J)=GGL(J)-YSL(I,J)*TEMP
  707 L=LS+1
      DO 710 I=1,LS
      L=L-1
      DO 712 J=NLS1,NLSV
  712 GGL(L)=GGL(L)-YSL(L,J)*VN(J)
      IF(LEQ.1) GOTO 710
      K=L+1
      DO 714 J=K,LS
  714 GGL(L)=GGL(L)-YSL(L,J)*GGL(J)
  710 CONTINUE
C-----------------------------------------------------------------
C       BACKWARD  COMPUTATION
C-----------------------------------------------------------------
  799 L=0
      DO 716 N=1,INC
      M=MCO(N)
      IF(M.LT.0) GOTO 720
      L=L+1
      TEMP=GGL(L)
      DO 718 J=1,M
      JJ=NCO(N,J)
  718 VN(JJ)=TEMP
      GOTO 716
  720 M=-M
      JJ=NCO(N,M)
      TEMP=VN(JJ)
      DO 722 J=1,M
      JJ=NCO(N,J)
  722 VN(JJ)=TEMP
  716 CONTINUE
  740 CONTINUE
      DO 727 I=1,NL
  727 VN(I)=SI(I)
      L=NL+1
      DO 728 I=1,NL
      K=L
      L=L-1
      DO 728 J=K,NLSV
  728 VN(L)=VN(L)-YL(L,J)*VN(J)
      IF(IVD.EQ.0) GOTO 730
      DO 732 I=1,IVD
      K=NODV(I)
      M=NVD(I)
  732 VN(K)=VN(K)+VN(M)
  730 CONTINUE
```

```
      IF(NC4.EQ.0) GOTO 744
      I=NC3+1
      DO 742 L=1,NC4
      K=IOUT(L,2)
      M=IOUT(L,3)
  742 OUT(L+I)=VN(K)-VN(M)
  744 CONTINUE
      RETURN
      END
```